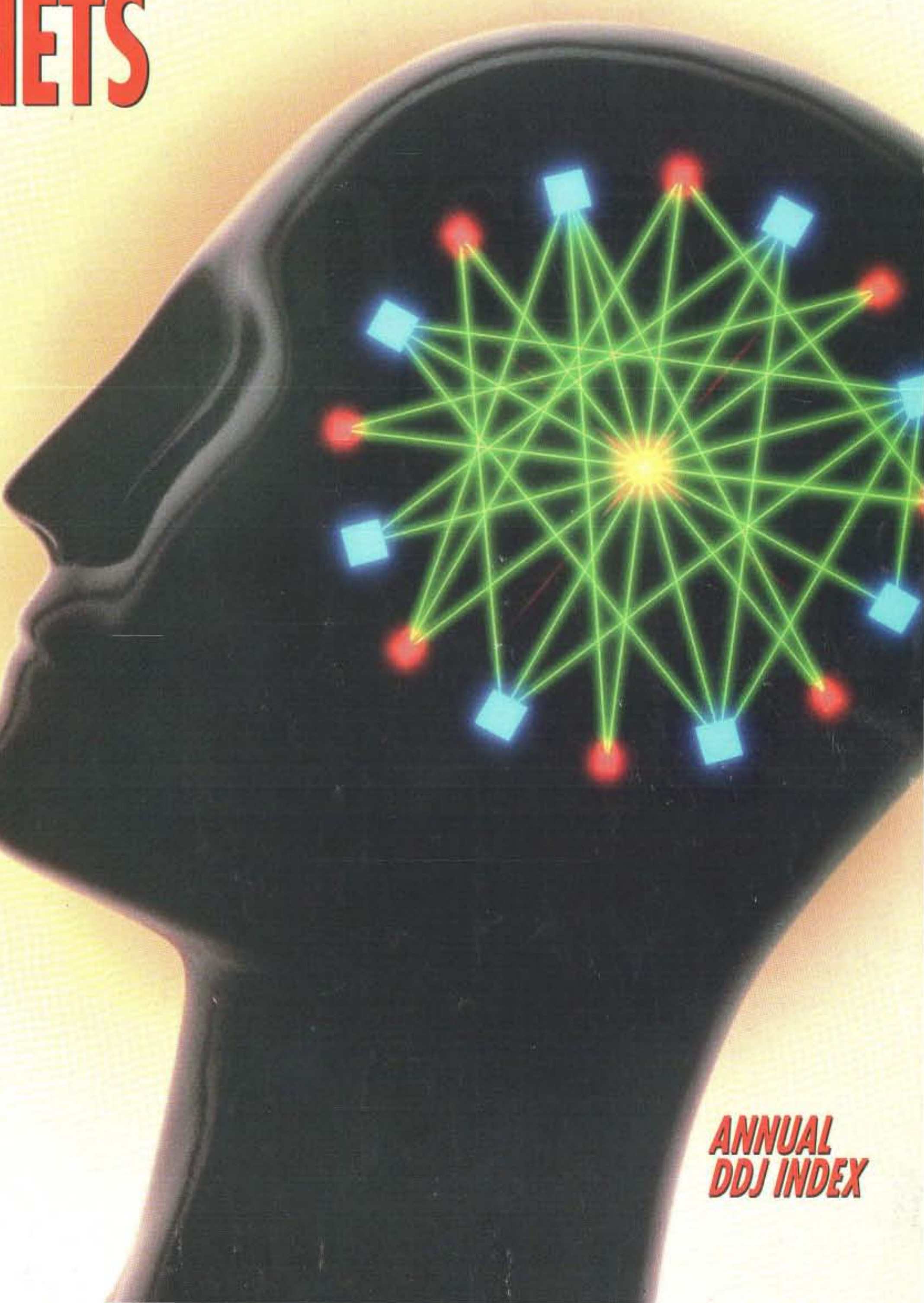
15 YEARS OF POWER PROGRAMMING

#163 APRIL 1990 $2.95 (3.95 CANADA)

Dr. Dobb's JOURNAL

SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER



NEURAL NETS NOW!

- BAM IN C++
- MODULARIZING NEURAL NETS
- IMAGE PROCESSING

VESA VGA

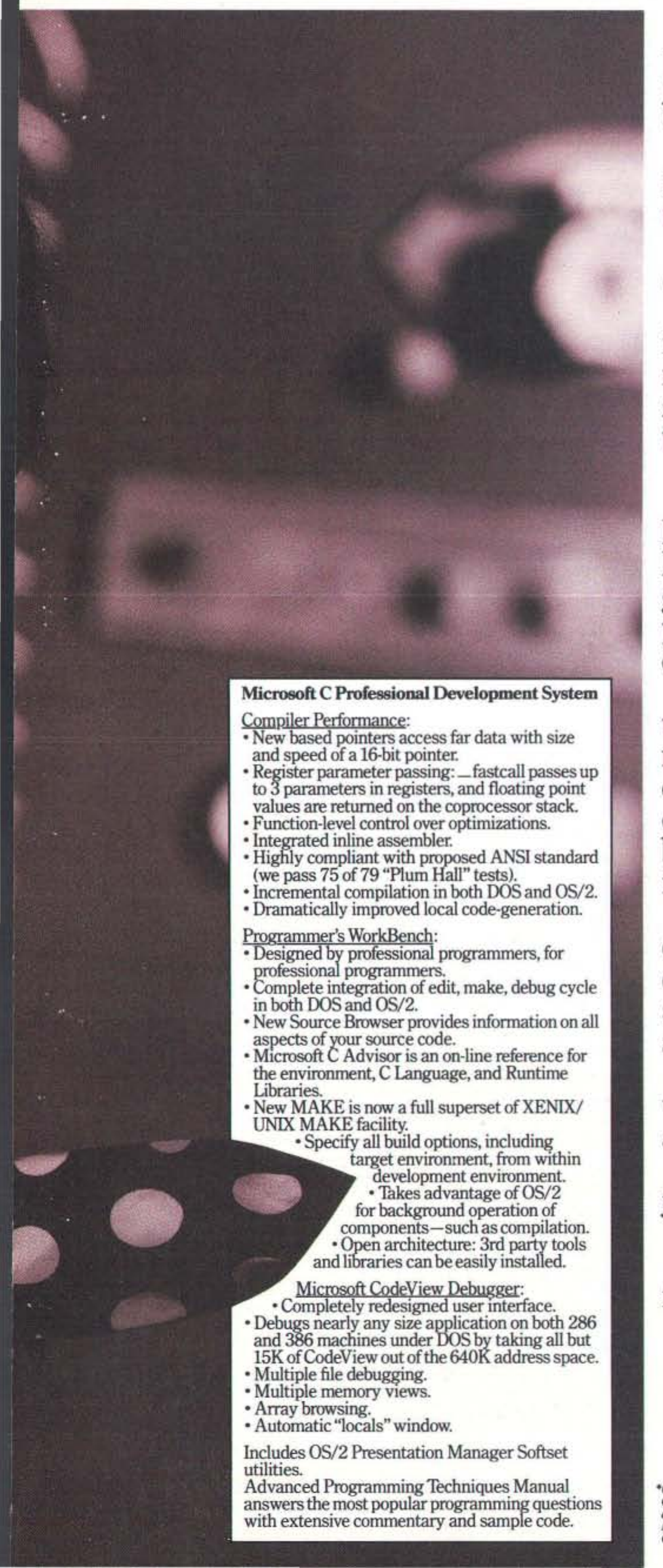
IMPLEMENTING RHEALSTONE

DATA COMPRESSION

ANNUAL DDJ INDEX

New C 6.0. It's not

for tachophobiacs.



If you love the rush of adrenalin that comes with sudden acceleration, we've got a C that's just your speed: the exhilarating new Microsoft® C 6.0 Professional Development System.

Designed to get your blood racing in nothing flat with the fastest,* slickest code available in the shortest, smoothest time possible.

Thanks to some of the most sophisticated code optimizations around.

With this C, you get everything from register-based parameter passing to a globally optimizing code generator to complete loop optimizations. Plus new super-efficient based pointers that access far data instantly.

And your code isn't all that moves faster.

Whether you're developing for MS-DOS,® Microsoft Windows™ or MS® OS/2 Presentation Manager systems, our brilliantly integrated Programmer's WorkBench propels you through the creative process with precision tools.

Including a new souped-up CodeView® Debugger that not only debugs any size DOS or OS/2 application program, on any 286 or 386 machine, but also gives you data-browsing so you don't have to constantly guess at values.

To save even more time, our C Advisor is on-line and on-time whenever you need it. This complete hypertext reference not only gives you sample coding solutions, but it even lets you copy and paste them directly into your program.

If you want to review part of your handiwork, our Source Browser helps you find it with a full call tree that literally draws you a map.

For a free white paper with more details, just call us toll-free at (800) 426-9400.

Then get your hands on Microsoft C 6.0.

The development system specially created for people whose only fear is being left behind.

Microsoft C Professional Development System

Compiler Performance:
- New based pointers access far data with size and speed of a 16-bit pointer.
- Register parameter passing: _fastcall passes up to 3 parameters in registers, and floating point values are returned on the coprocessor stack.
- Function-level control over optimizations.
- Integrated inline assembler.
- Highly compliant with proposed ANSI standard (we pass 75 of 79 "Plum Hall" tests).
- Incremental compilation in both DOS and OS/2.
- Dramatically improved local code-generation.

Programmer's WorkBench:
- Designed by professional programmers, for professional programmers.
- Complete integration of edit, make, debug cycle in both DOS and OS/2.
- New Source Browser provides information on all aspects of your source code.
- Microsoft C Advisor is an on-line reference for the environment, C Language, and Runtime Libraries.
- New MAKE is now a full superset of XENIX/UNIX MAKE facility.
- Specify all build options, including target environment, from within development environment.
- Takes advantage of OS/2 for background operation of components—such as compilation.
- Open architecture: 3rd party tools and libraries can be easily installed.

Microsoft CodeView Debugger:
- Completely redesigned user interface.
- Debugs nearly any size application on both 286 and 386 machines under DOS by taking all but 15K of CodeView out of the 640K address space.
- Multiple file debugging.
- Multiple memory views.
- Array browsing.
- Automatic "locals" window.

Includes OS/2 Presentation Manager Softset utilities.

Advanced Programming Techniques Manual answers the most popular programming questions with extensive commentary and sample code.

Microsoft®
Making it all make sense™

Comprehensive benchmark tests performed at Microsoft on an 8Mhz IBM PC/AT with 2816K RAM and 80287 coprocessor. Customers within the 50 United States, call (800) 426-9400. Customers in Canada, call (416) 673-7638. Outside the U.S. and Canada, call (206) 882-8661. © 1990 Microsoft Corporation. All rights reserved. Microsoft, the Microsoft logo, MS-DOS, MS and CodeView are registered trademarks and Windows and Making it all make sense are trademarks of Microsoft Corporation. PC/AT is a registered trademark of International Business Machines Corporation.

WE HAVE WAYS TO MAKE YOU TALK

(and let you listen in, too)

with tools from Blaise Computing.

Talk to your modem and connect with remote systems.

Go online with ease with our new high-level modem control routines. Initialize, dial, answer, auto-answer, and auto-adjust for incoming baud rate.

Let's talk file transfer.

XModem/YModem routines let you send and receive multiple files over multiple ports. If your program has ISR/TSR capability (as provided by our POWER TOOLS PLUS, C TOOLS PLUS, or Turbo C TOOLS), you can talk file transfer in the background while your foreground process talks data input, or database query, or whatever.

The file transfer capabilities include 1K packets, CRC error checking, YModem (multi-file transfers with file name and size preserved), automatic adaptation to incoming packet size and error detection method. And we're talking fast—all file transfer routines have been optimized for maximum throughput.

Let's talk basics.

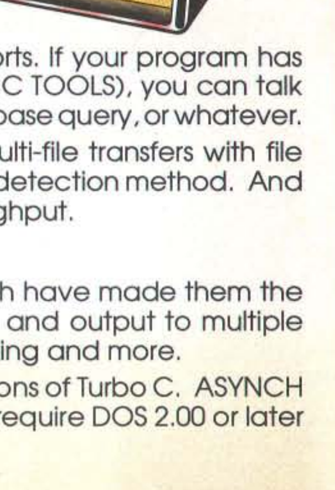
C ASYNCH MANAGER™ and **ASYNCH PLUS™** retain the tried and true features which have made them the libraries of choice for completing asynch projects—buffered interrupt-driven input and output to multiple COM ports with speeds up to 19200 Baud, XON/XOFF protocol, hardware handshaking and more.

C ASYNCH MANAGER supports Microsoft C version 4.0 and later, QuickC and all versions of Turbo C. ASYNCH PLUS supports Turbo Pascal versions 4.0 and later and QuickPASCAL. Both products require DOS 2.00 or later and an IBM PC, XT, AT or PS/2 machine or true compatible.

Let's talk price.

Why pay more? Our ASYNCH packages are just $189, including complete source code, a comprehensive reference manual with extensive examples, sample programs and online help.

Turn that serial cable into a party line with VIEW232!



Debugging serial communications doesn't have to leave you talking to yourself. New from Blaise computing: **VIEW232™** turns your computer into a serial data analyzer that lets you listen in on the conversation between any two serial devices —and this party line provides a complete transcript of what was said!

VIEW232 displays the data as it flows in both directions. And you can set the display format (6 formats supported) and trap predetermined character sequences. Save a whole transmission in a buffer, then browse through or search the buffer for a pattern or specific character. And you can save the data to disk or print it for later study.

VIEW232 is easy to use. And we supply the cable, all for just $189!

Let's talk track record.

Blaise Computing has produced a collection of tools over the years that are unsurpassed for reliability, flexibility, and ease of use. Call today for a free brochure on our complete product line.

And we're talking guaranteed.

We offer a 30-day money back guarantee. If during the first 30 days you are not completely satisfied, we'll refund your money.

BLAISE COMPUTING INC.

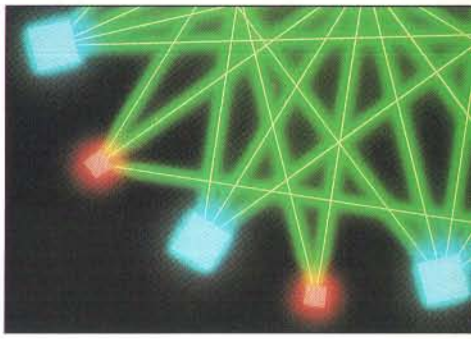
2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441 **CIRCLE NO. 91 ON READER SERVICE CARD**

(800) 333-8087 **Price per package $189!** FAX (415) 540-1938

All trade names are the trademark or registered trademark of the respective manufacturer.

CONTENTS

APRIL 1990
VOLUME 15, ISSUE 4

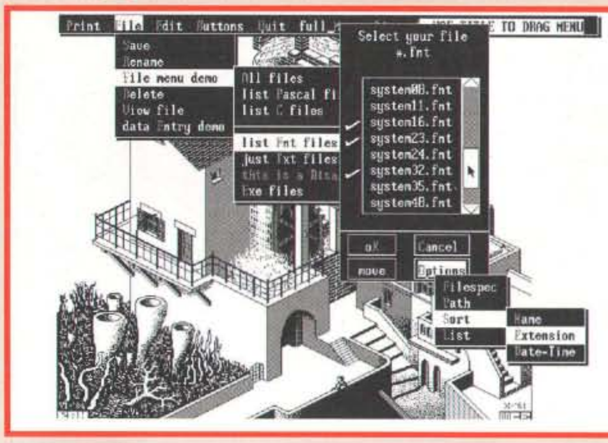


NEXT ISSUE

Bigger programs can lead to bigger memory management headaches. In May, we'll examine different approaches to memory management and continue our in-depth coverage of 80386 programming.

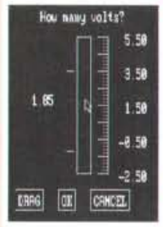
The professional menu system in a *graphics* environment
for TURBO C, MICROSOFT C, TURBO PASCAL

***graphics*-MENU**



***graphics*-MENU**™ from ISLAND SYSTEMS is a comprehensive library plus utilities package that allows the developer of graphics applications to quickly create a user-friendly interface with menus, messages and forms.

- Several types of menus and message boxes.
- Mouse support functions and geometric functions such as point rotation, distance, degrees/radians conversion, true 4-quadrant arctan.
- EMS support
- Foreground/ background task capability
- Automatically saves/restores underlying graphics image
- Shadows for 3d effect

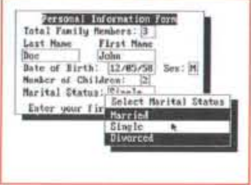


- Most items draggable by mouse
- Menus can work with mouse or keyboard or both.
- **CASE tools: MenuDesigner**™ **and Data Entry Designer**™ (opt. add-ons) provide on screen WYSIWYG design environments for menu creation and include automatic code generation.
- **Utilities** *included in the base package*: **CurEdit™** creates custom cursor icons; **ColorCustomizer**™ allows complete color specification via an on-screen pallette.

Data Entry

The ***Data Entry Module***™ is a specialized library (optional add-on to *graphics*-MENU) to enable the developer to easily create forms type menus. A high degree of flexibility has been built-in to allow "hooks" into the library for all types of data entry applications.

- dBASE-like string specification
- real-time data checking
- data type validation
- automatic field movement
- several types of form exit conditions
- help message facility
- professional string editing


Personal Information Form
Total Family Members: 3
Last Name Doe
First Name John
Date of Birth: 12/05/58 Sex: M
Member of Children: 2
Marital Status: Single
Enter your fir

- **CASE tool: Data Entry Designer** (optional) available to quickly create complex forms menus on-screen with auto code generation.
- Price for Data Entry Module with Data Entry Designer just $99! Source Available.

graphics-MENU works with MetaWindow (from Metagraphics Software) or BGI (Borland Graphics Interface).
Language Versions available for Turbo C, Microsoft C, Turbo Pascal.

Base package price is $99 (BGI versions)
$149 (META versions)
Several add-on options and source available.
Call for additional information or pricing.
We accept Mastercard and VISA
**** 30 day money back guarantee ****

CALL: (617) 273-0421
FAX: (617) 933-1152
DEMO AVAILABLE

Island Systems

7 Mountain Rd. Burlington MA 01803

CIRCLE NO. 304 ON READER SERVICE CARD

Dr. Dobb's JOURNAL SOFTWARE TOOLS FOR THE PROFESSIONAL PROGRAMMER

PUBLISHER *Peter Hutchinson*

EDITORIAL
EDITOR-IN-CHIEF *Jonathan Erickson*
MANAGING EDITOR *Monica E. Berg*
TECHNICAL EDITORS *Michael Floyd, Ray Valdes*
EDITORIAL ASSISTANT *Janna Custer*
CONTRIBUTING EDITORS *Al Stevens, Jeff Duntemann, Martin Tracy, David Betz, Tom Genereaux, Andrew Schulman*
COPY EDITORS *Rhoda Simmons, Pamela Dillehay, Nan Fornal*
EDITOR-AT-LARGE *Michael Swaine*

ART/PRODUCTION
ART/PRODUCTION DIRECTOR *Larry L. Clay*
ART DIRECTOR *Michael Hollister*
PRODUCTION SUPERVISOR *Amy Shulman Lesovoy*
TYPOGRAPHERS *Teresa Raines, Margaret Anderson, Charlene Carpentier*
COVER PHOTOGRAPHER *Michael Carr*

CIRCULATION
DIRECTOR OF CIRCULATION *Maureen Kaminski*
CIRCULATION MANAGER *Randy Robertson*
CIRCULATION PLANNING MANAGER *Manny Sawit*
DIRECT MARKETING MANAGER *Andrea Weingart*
NEWSSTAND MANAGER *Sarah Forsman*
DIRECT MARKETING COORDINATOR *Francesca Davies*
PROMOTION COORDINATOR *Pam Moore*
FULFILLMENT COORDINATOR *Anne Jean*

ADMINISTRATION
VICE PRESIDENT OF FINANCE *Kate Deschamps*
CONTROLLER *Mary Collopy*
CREDIT MANAGER *Betty Arsene*
ACCOUNTING SUPERVISOR *Renate Kernke*
ACCOUNTS RECEIVABLE *Wendy Ho*
ACCOUNTS PAYABLE *LuAnn Rocklewitz*

MARKETING/ADVERTISING
DIRECTOR OF SALES AND MARKETING *Karla Spormann*
ADVERTISING COORDINATOR *Laura Stack Pullen*.
MARKETING ASSISTANT *Sara Noah Ruddy*
ACCOUNT MANAGERS *see page 153*

M&T PUBLISHING INC.
CHAIRMAN OF THE BOARD *Otmar Weber*
DIRECTOR *C. F. von Quadt*
PRESIDENT *Laird Foshay*
VICE PRESIDENT OF PUBLISHING *William P. Howard*
VICE PRESIDENT/GROUP PUBLISHER *Randall L. Stickrod*

DR. DOBB'S JOURNAL (USPS 307690) is published monthly, except semimonthly in December, by M&T Publishing, Inc., 501 Galveston Dr., Redwood City, CA 94063; 415-366-3600. Second-class postage paid at Redwood City and at additional entry points.

ARTICLE SUBMISSIONS: Send manuscripts and disk (with article and listings) to the editorial assistant 415-366-3600.

***DDJ* ON COMPUSERVE:** Type GO DDJ.

***DDJ* LISTING SERVICE:** 603-882-1599. Supports 300/1200/2400 baud, 8-data bits, no parity, 1-stop bit. Type *listings* (use lowercase) at the login prompt.

SUBSCRIPTION: $29.97 for 1 year; $56.97 for 2 years. Foreign orders must be prepaid, including the additional postage (air or surface) in U.S. funds drawn on a U.S. bank. Add $13 per year for surface mail; add $36 for airmail to Canada and Mexico; or $26 for airlift to all other countries.

POSTMASTER: Send address changes to *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80322-6188. **ISSN 1044-789X**

CUSTOMER SERVICE: For subscription questions, call toll-free 800-456-1215. For subscription orders or change of address call toll-free 800-456-1215 (U.S. and Canada) or write *Dr. Dobb's Journal*, P.O. Box 56188, Boulder, CO 80322-6188. For book/software orders call 800-533-4372 (in California 800-356-2002).

FOREIGN NEWSSTAND DISTRIBUTOR: Worldwide Media Service Inc., 115 E. 23rd St., New York, New York 10010; 212-420-0588 FAX 212-420-1265.

C Network Compiler.
C Network Compiler Run.
C Network Compiler Run 386.

Introducing C Network Compiler/386.

Introducing another first from Novell–the first network compiler for developing distributed applications.

C Network Compiler/386 gives you the fastest, most streamlined code you've ever seen, running on NetWare® 386, the fastest, most efficient network operating system we've ever developed.

C Network Compiler/386 is a complete and integrated set of NetWare 386 programming tools that generates 32-bit, native-mode 386 code. Now you can use a standard programming tool to conquer the complexities of building server applications.

And for building client applications, we offer the DOS version of C Network Compiler for the entire 80x86 processor family. Both compilers are based on the latest technology from WATCOM Systems Inc., and are optimized for use with NetWare.

Besides the powerful ANSI C compiler, enhanced debugger, linker, and other utilities, with our network compilers you have the whole library of NetWare application programming interfaces, including Btrieve®, at your command.

That gives you a direct link into NetWare, the leading network operating system with the world's largest installed base of network application users.

Call us and see how fast our C Network Compilers can make your applications run. And start writing applications today that will satisfy the network computing needs of the next decade.

C Network Compiler — US$695
C Network Compiler/386 — US$995

NOVELL

For network solutions, you should be seeing red.

Call before you write. 512-346-8380

It Takes More Than Nerve

The biggest fear of those who champion neural networks is guilt-by-comparison with the artificial intelligence camp. They're not alone in this. Object-oriented advocates, as well as most other popular technologies that make the front pages of pseudo-technology news tabloids, don't want to be snake-bit by the same type of hype that poisoned AI development. The frontal assaults of AI and expert systems, fueled by big money and bigger promises, have been nonexistent for neural nets and, although those neural net developers trying to eke out a living might disagree, the lack of venture capital has probably been a blessing. Less hype buys more time, at least as long as enough money comes in to keep the lights turned on.

Nevertheless, there continues to be a lot of interest and development in neural nets. A survey recently published by Future Technology Surveys of Madison, Georgia listed over 200 companies and organizations currently producing neural-related products or undertaking serious neural net research. And it just isn't the little guys doing all this research, either.

Among the big outfits testing the neural net waters is Intel. Early last summer, a ten-member Intel engineering team, under the direction of Mark Holler, rolled out an Electronic Trainable Artificial Neural Network (ETANN) chip that is capable of up to 2 billion multiplies and accumulates per second. To put the chip to work, Holler and his crew have built a prototype ETANN-based board that plugs into the PC AT bus; Mark Lawrence and the folks at California Scientific (developers of BrainMaker, a neural net simulation package for PCs) are developing the software tools that let you use the system. There's also a rumored Intel research project that will put a version of the ETANN board into an i860-based system that can achieve 33 billion connections per second.

Intel isn't the only big IC manufacturer poking around in neural nets. A few months ago, Sharp introduced a neural-network image-processor chipset that simulates human vision and, the company claims, supports PC applications at speeds up to 700 MIPS.

These examples illustrate another trend in the neural net world — a transition from software to hardware. Within ten years, or so say the experts, more neural nets will be implemented in hardware than software. Until then, engineers will begin to overcome many challenges, including the implementation of back propagation in hardware and the parallelization of the entire scheme.

So where does this leave software developers? For one thing, a whole new class of development tools is in the offing, designed for specific neural net hardware implementations. Another type of tool will be like that described by Andy Czuchry in this issue, whereby designers can match the right neural model with the task at hand. Nor will the simulators go away; they may be used to simulate the right net with appropriate learning, then generate source code to be frozen in silicon.

In her keynote address at Miller-Freeman's SD'90, Smalltalk pioneer and ParcPlace System's president Adele Goldberg expressed a concern similar to that I wrote about in this space last month — the spread of litigation and its effect on the software industry.

Although her talk concerned a wide variety of legal issues — from intellectual property to the emerging problem of who owns the design and implementation of objects, as in object-oriented programming — she spent a fair amount of time on copyrights and patents. "Lawyers will always tell you two things," she said, "try to patent or copyright whatever you do." She went on to describe a speech she gave to a group of lawyers, where she was asked how to convince software developers to protect their works. "My answer was simple," she said. "Tell them to protect their work so that they have the choice later on to give it away." Not doing so, she explained, opens the door for someone to come along and take it away. But Goldberg wasn't engaged in lawyer-bashing, no matter how easy that is. What she was presenting was a persuasive argument for open standards and open licensing. She pointed out that among the problems litigation forces upon us are the waste of time and money, the fear of alliances, the inability of entrepreneurs who lack clear patent or copyright protection to attract investors, and the expense of starting up new businesses.

One of the main points of her talk was simply to "reassert an often unstated goal of our industry — to share ideas and to challenge one another with our innovative expressions of those ideas." Nicely put.

And no, the favored horse running in the first race at Bay Meadows racetrack the other night wasn't our official mascot, even though the nag's name was "Dr. Dobbs." Although, he lost by a nose in a photo finish, the good Doctor is surely chomping at the bit to get into the next race. We'll keep you posted on his progress this season.

Jonathan Erickson

Jonathan Erickson
editor-in-chief

QNX®
The OS for over-achievers™

QNX programmers have a decided advantage.

You see, people who use QNX enjoy the freedom that comes only with a flexible, modular OS. They appreciate the elegance of a **message-passing architecture**. And they marvel at the fact that QNX runs so lean — under 150K — yet out-performs any other PC operating system.

QNX users never worry about whether their applications will make it at runtime, because they know QNX has proven itself again and again in the real world.

It's no wonder that QNX users have achieved so much since the product was first released for the PC in 1982: over 80,000 systems installed in 47 countries world-wide, in all kinds of applications — from making cars to selling books to handling online credit card transactions.

One reviewer dubbed QNX "The multi-everything OS." Now, you might expect multiuser and multitasking, but realtime? *And* integrated networking? *And* true distributed processing? Best of all, these terms take on a new meaning with QNX.

Multiuser, for instance, means up to 32 terminals per micro. **Multitasking** cashes out as 150 tasks per machine. **Realtime** means not only priority-driven, preemptive task scheduling, but also speed: at 6,896 task switches/sec on a 16MHz 286, QNX is at least a full order of magnitude faster than a typical UNIX system. **Integrated networking** means you won't need yet another layer of software to set up a LAN, and you can use *any mix* of Intel-based micros — from vintage '81 PCs to PS/2s.

Distributed processing with QNX sounds too good to be true. But it is: *Any task can access any resource* — programs, files, devices, even CPUs — without going through the bottleneck of a central file server.

Besides the satisfaction that QNX developers get from using a fast, powerful, and flexible OS, did we mention that they also enjoy *free technical support?*

If you're wondering why you don't already know all about this great OS, you could try asking the over-achievers who are smugly guarding the secret of their success.

Better yet, give us a call. We'll tell you everything you need to know to become an over-achiever yourself.

QNX

For more information or a free demo disk, please phone (613) 591-0931.

Quantum Software Systems Ltd., 175 Terrence Matthews Crescent, Kanata, Ontario, Canada K2M 1W8

Faster Animation

Dear *DDJ*,

I enjoyed Rahner James's article on "Real Time Animation" in the January 1990 issue of *DDJ*. Ever since the price of EGA devices dropped, magazines have been filled with how-to articles, but few have covered icons or sprites. In 1986 I ported a collection of graphics routines from my Zenith 100 system to the EGA. These routines are now part of an icon development environment called "ProGraphx Toolbox," available from Stanwood Associates.

Speed is definitely the key to success in graphics, and Mr. James's routines definitely are fast. In my routines I have come up with another approach, which, I believe, is slightly faster in displaying icons. Since the EGA is latched, you must determine which planes will be accessed during a write, a clock cycle consuming process. Mr. James's routines store one byte per pixel, enabling 128 colors and an intensity bit. If the format of the sprite were laid out plane by plane rather than pixel by pixel, the function could set the registers for a plane and then write all the data for that plane. The function would continue plane by plane, only setting up the register once per plane per icon. Additionally, the EGA does not allow bit access. So why not place a byte's worth of data on the screen during each write rather than only one new bit? I enjoy seeing quality articles every time I open a new issue of *DDJ*. Keep it up!

Peder Jungck, Stanwood Assoc.
Chicago, Illinois

On Location

Dear *DDJ*,

The excellent article "Location is Everything," by Mark Nelson, (January, 1990) was most timely. Once again *DDJ* came up with just what I wanted just when I needed it. I think, however, there is a small problem with the code.

Mark uses the exe header field "Displacement of stack in Paras" (0e) to locate the start of the initialized data area. This works only when the amount of initialized data is less than one paragraph long since the stack displacement corresponds to the end of the initialized data area. In the general case, the program needs the starting paragraph. I was able to easily find this value by parsing it out of the .MAP file. Using this value in the relocation function causes the program to perform as advertised.

```
/* input_base_data_segment is a global
                              unsigned int */
/* data_seg is declared as char
                              data_seg[81] */
/* map_file is a FILE * to the .MAP file
                              created by */
/* Scan the .map file for the word "BSS"
                                        */
while(strcmp(data_seg, "BSS") ! = 0)
      fscanf(map_file, "%s", data_seg);
/* The next field in the file is the loca-
                              tion of the */
/* data. */
fscanf(map_file, "%s", data_seg);
data_seg[strlen(data_seg)-1] = '\0';
/*kill the 'H' */
input_base_data_segment =
                        htoi(data_seg) >> 4;
```

In the function *process_relocation_table( )*, replace all references to the variable *first_data_segment_in_exe_file* with *input_base_data_segment.*

Stephen J. Beaver
Winchester, Virginia

Mark responds: As Stephen Beaver notes, there is a field in the header portion of an EXE file that tells me where the start of the program stack segment is located. I use this field to determine where RAM data starts in the EXE file. Mr. Beaver must have taken note of the lines in my START.ASM file shown below. Because the stack segment follows the DATA, BSS, and CONST segments, Mr. Beaver concludes that the value I calculate for the start of RAM actually points past all these segments. However, there is an additional segment definition line a little farther down in the START.ASM file:

```
DGROUP    GROUP _CONST, _BSS,
                    _DATA, _STACK
```

This statement causes the compiler and linker to gather all four of these segments together into one segment. This means that all four segments are collected together, and the pointer to the start of the stack segment will actually point to the start of DGROUP, which will be at the start of the _CONST segment. So, if you use the START.ASM startup file as I required, the LOCATE.EXE program will work properly. By the way, Mr. Beaver "solved" this problem by modifying my LOCATE program to read in a MAP file that the linker has produced. Reading in MAP files to drive a Locate program is not a bad approach. In fact, Jensen & Partners International are providing a TSLOCATE utility with their new TopSpeed C compiler, which does just that. I chose to avoid this approach for a couple of reasons. First, there is no standard MAP file format. Every linker is free to create their own format, and a good LOCATE program would be forced to continually adapt to these. Second, a program using this approach is vulnerable to errors caused when the MAP file is not actually the one from the latest link. Because the information I wanted was in the EXE file, and the EXE format is standardized across all MS-DOS compilers, I elected to not read in the MAP file. *I hope this clears up some of the confusion. Dealing with program segments at a low level is usually concealed from HLL programmers by the compiler, for which we can all give thanks.*

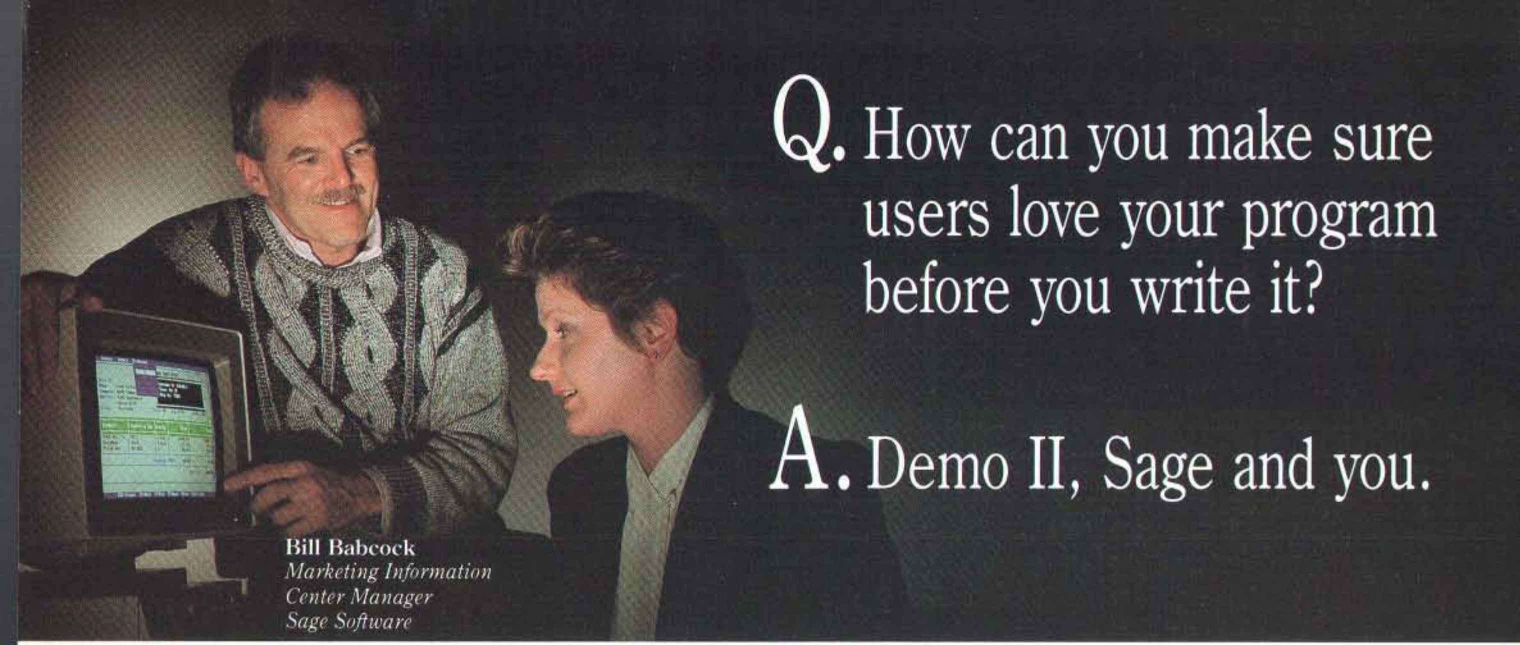
Random Structures

Dear *DDJ*,

This is in response to the December 1989 letter by Dan W. Crockett. He is treating the term "structured" and the term "modular" as being equivalent. A structured module, program, or system

(continued on page 12)

```
_TEXT          SEGMENT BYTE PUBLIC 'CODE'
_TEXT          ENDS
END_OF_ROM     SEGMENT PARA PUBLIC 'STARTUP_CODE'
END_OF_ROM     ENDS
_CONST         SEGMENT PARA PUBLIC 'CONST'
_CONST         ENDS
_BSS           SEGMENT WORD PUBLIC 'BSS'
_BSS           ENDS
_DATA          SEGMENT WORD PUBLIC 'DATA'
_DATA          ENDS
_STACK         SEGMENT WORD STACK 'STACK
MYSTACK        DB 512 DUP (?)
_STACK         ENDS
```


Q. How can you make sure users love your program before you write it?
A. Demo II, Sage and you.

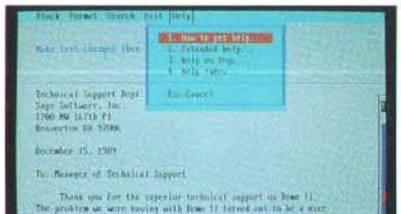
Bill Babcock
Marketing Information Center Manager Sage Software

“True story. I'm developing a marketing info system at Sage. Big Project. I've been working with the users for months. I've got notebooks full of data structures, screen drawings, report requirements, and menu designs.

“We're two weeks into coding when my boss gives me a copy of Dan Bricklin's® Demo II.™ 'We're going to acquire this,' he says. So I play with it over the weekend. I prototype my project. On Monday I proudly demo my prototype to the users who *completely* agreed with my paper plans.

“They hate it. Vital information is not accessible from screens where it's needed; frequently used functions are hidden deep in menus; the entry flow doesn't fit the work flow. I don't know whether they did a bad job of telling me what they wanted or I did a bad job of listening.

“But with a few hours of work I fix the problems in the prototype. They try it again. Big smiles. The project is back on track. They love it.


Block Format Search Exit Help
1. How to get help...
2. Extended help...
Esc Cancel
TYPING MENU
Lines Chars Direction HTabs VTabs Margin Status Find
Make cursor keys draw lines

```
Technical Support Dept
Sage Software, Inc
1700 NW 167th Pl
Beaverton OR 97006

December 15, 1989

To: Manager of Technical Support

     Thank you for the superior technical support on Demo II.
The problem we were having with Demo II turned out to be a user
```

Demo II can build interactive demos so realistic that users can't tell them from finished software. Above is an example of a CUA interface design that will be running on both mainframe terminals and PS/2s.

“The same hands-on demo with real code would have happened months into the project. We would have trashed a lot of work or finished with a system nobody wanted. Demo II saved my project and my time. Now I know why it's so popular.”

Dan Bricklin's Demo II can do more than just save you time and trouble. Use it to provide astonishingly realistic demos of applications - self running, interactive, or both. Use it to show management and users that your project is on track. Use it whenever you need to communicate how a program will look, feel, and respond without providing the real code. And though Demo II runs on MS-DOS, many programmers use it to create prototypes for mainframe and minicomputer applications.

Even non-programmers find it easy to learn, but Demo II has so much capability and stores demos so compactly that it's widely used for computer-aided instruction, replacing CAI authoring systems that cost thousands of dollars.

Don't wait until your carefully researched and brilliantly coded project turns out to do the wrong thing the wrong way. The time, the grief, and the reputation you'll save are your own.

Polytron has been acquired by Sage Software, Inc. The same team that supports and develops the Polytron products you are familiar with, now supports Demo II.

DEVELOPMENT POLYTRON TOOLS

$199 U.S. List Price with unlimited run-time distribution rights.

1-800-547-4000

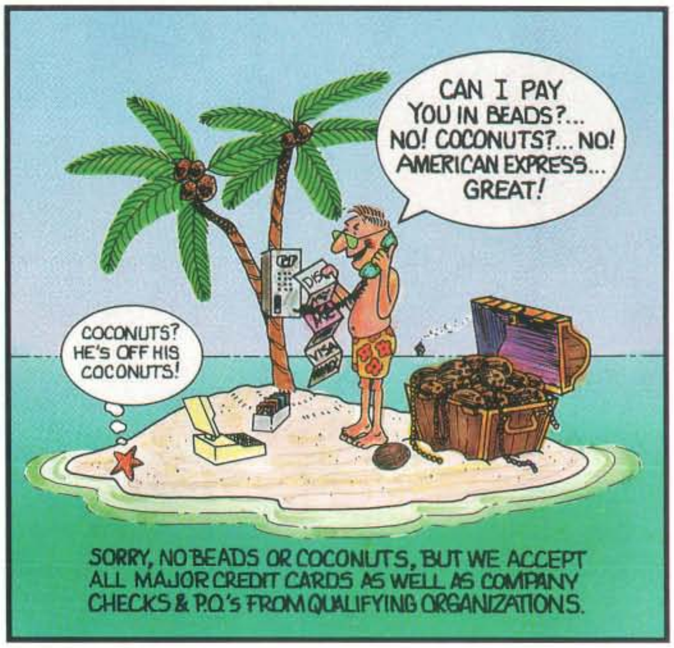
30 Day Money Back Guarantee

SAGE SOFTWARE

Dan Bricklin's is a reg. trademark of Daniel Bricklin. Requires - 512K IBM PC compatible, Mono., CGA, EGA, VGA or Hercules. MS-DOS 2.0 or later.
Sage Software, Inc., 1700 NW 167th Place, Beaverton, OR 97006, (503) 645-1150, Fax (503) 645-4576, Int'l. (301) 230-3307.

CIRCLE NO. 191 ON READER SERVICE CARD

Programmer's Paradise... for


CAN I PAY YOU IN BEADS?... NO! COCONUTS?... NO! AMERICAN EXPRESS... GREAT!
COCONUTS? HE'S OFF HIS COCONUTS!
SORRY, NO BEADS OR COCONUTS, BUT WE ACCEPT ALL MAJOR CREDIT CARDS AS WELL AS COMPANY CHECKS & P.O.'s FROM QUALIFYING ORGANIZATIONS.

WE'LL MATCH NATIONALLY ADVERTISED PRICES.

	LIST	OURS
386 CONTROL PROGRAMS		
DESQview 386	190	169
Microsoft Windows/386	195	139
VM/386	245	199
VM/386 Multi-User	895	819
386 DEVELOPMENT TOOLS		
386 ASM/LINK	495	435
Lahey F77L-EM/32 (w/ OS/386)	1090	975
Novell C Network Compiler/386	995	779
Paradox/386	895	629
WATCOM C 7.0/386	895	799
ASSEMBLY LANGUAGE		
Advantage Disassembler	295	279
ASMFlow	99	89
ASMTool	90	80
MS Macro Assembler	150	105
OPTASM	125	109
Re:Source	150	129
Sourcer w/ Pre-Processor	140	125
Turbo Assembler/Debugger	150	105
Visible Computer: 80286	100	89
BASIC COMPILERS		
MS BASIC Prof. Devel. System	495	349
Power Basic	110	99
QuickBASIC	99	69
True BASIC	100	69
BASIC LIBS/UTILITIES		
db/LIB	139	121
DiaLogic	79	70
GraphPak	79	70
GraphPak Professional	149	125
LaserPak	79	70
P.D.Q.	99	89
ProBas	135	125
ProBas HyperHelp Toolkit	99	94
ProBas Telecomm. Toolkit	75	70
ProBas Toolkit	99	94
ProMath	99	94
ProScreen	99	89
QBase and Quickscreen	149	125
QuickComm	139	125
QuickMenu	59	55
QuickPak	79	70
QuickPak Professional	149	125
QuickPak Scientific	79	70
QuickScreen	79	70
QuickWindows Advanced	149	125
QuickWindows Advanced Corp.	500	445
C COMPILERS		
C Network Compiler	695	525
Lattice C 6.0	250	155
Microsoft C 6.0	CALL	CALL
MS Quick C	99	69
MS QuickC w/ QuickAssembler	199	139
Top Speed C	199	179
DOS Professional	399	359
OS/2 Professional	495	445
Turbo C	150	99
Turbo C Professional	250	159
WATCOM C 7.0	395	319

	LIST	OURS
C++		
Guidelines C++	295	269
NDP C++	495	479
Zortech C++ Debugger	150	129
Zortech C++	200	165
Developer's Edition	450	399
Zortech C++ Tools	150	129
Zortech C++ Video Course	500	449
C-COMMUNICATIONS		
Breakout II	125	99
C Asynch Manager 3.0	189	139
Essential Communications	329	259
Greenleaf Comm. Library	299	215
Greenleaf ViewComm	559	475
SilverComm C Async Library	249	209
View-232	189	149
C-FILE MANAGEMENT		
Btrieve	245	185
Btrieve for DOS 3.1 Networks	595	449
C-Index Plus	195	175
C-ISAM	225	209
Codebase IV	295	219
CQL w/ PASS	395	349
c-tree	395	315
dBC III	250	219
dBC III Plus	500	439
db_FILE Bundle	295	249
Essential B-Tree w/ source	199	149
FairCom Toolbox - Prof. Edition	1095	789
FairCom Toolbox - Special	695	509
Informix Products	CALL	CALL
Xtrieve PLUS	595	459
C-GENERAL LIBRARIES		
C TOOLS PLUS/6.0	149	109
C Utility Library	249	175
Greenleaf Functions	229	159
Greenleaf SuperFunctions	299	209
Power Search	149	99
Turbo C TOOLS/2.0	149	109
C SCREENS		
C-Worthy w/ forms and source	495	CALL
Greenleaf DataWindows	395	309
Hi-Screen XL	149	129
JAM	595	529
Panel Plus	495	395
Power Screen	149	109
Vermont Views	CALL	CALL
Vitamin C	225	165
VC Screen	149	115
C-UTILITIES/OTHER		
Clear +	200	169
C-Terp	300	219
Code Runner	149	135
Heap Expander	80	70
HyperWindows	99	90
Norton Guides for C	100	65
PC-lint	139	109
PCYACC Professional	495	469
TimeSlicer	295	279
w/ source	1000	899

	LIST	OURS
COBOL LANGUAGE		
Micro Focus:		
COBOL/2 w/ Toolset	1800	1499
Personal COBOL	149	129
MS COBOL	900	629
Realia COBOL	995	849
SCREENIO	400	375
CODE GENERATORS		
C Source	395	299
Logic Gem	99	89
Matrix Layout 2.0	200	169
PRO-C	399	339
DATABASE DEVELOPMENT		
Clarion 2.0	695	499
Clipper 5.0	695	519
dBASE IV	795	489
dBFast/PLUS	249	219
dGE	195	179
FlashTools!	89	79
FoxPro	795	635
Magic PC	299	249
R&R Report Writer	150	129
R&R Code Generator	150	129
Say What?!	50	45
SilverComm Library 2.0	189	165
Tom Rettig's Library	100	80
UI2 Version Two	595	479
DOCUMENTING/ FLOWCHARTING		
Clear+	200	169
C-Clearly	130	115
Flow Charting II+	229	185
Interactive Easyflow	150	125
Paginate	100	90
Source Print	99	89
The Documentor	295	245
Tree Diagrammer	99	89
EDITORS		
BRIEF 3.0	199	CALL
Edix	195	165
EMACS	325	265
Epsilon	195	138
KEDIT 4.0	150	125
MKS Vi	149	129
Multi-Edit	99	89
Multi-Edit Professional	179	159
Norton Editor	75	59
SLICK Editor	195	175
SPF/PC	245	199
VEDIT PLUS	185	115
Vq^2	150	135
FORTRAN LANGUAGE		
Grafmatic	135	119
Lahey F77L	595	529
Lahey Personal FORTRAN 77	95	89
MS FORTRAN	450	299
Plotmatic	135	119
RM/FORTRAN	595	499
GRAPHICS LIBRARIES		
Baby Driver	250	199
Essential Graphics	399	279
Font-Tools	150	119
Font Window	125	109
GraphiC 5.0	395	319
Graphics-MENU	195	175
Data Entry Design	99	89
Data Entry Module	59	53
GSS Graphics Devel. Toolkit	595	509
HALO	395	279
HALO Window Toolkit	595	419
Icon-Tools/Plus	150	119
Menuet	250	199
MetaWindow	250	209
MetaWindow Plus	325	269
PCX Effects	99	89
PCX Programmer's Toolkit	195	175
PCX Text	149	135
Turbo Geometry Library	200	179
LINKERS/LIBRARIANS		
Plink86plus	495	395
PolyLibrarian II	149	135
.RTLink	295	265
.RTLink/Plus	495	419
MODULA-2		
LOGITECH Modula-2:		
Compiler Pack	99	75
Development System	249	199
TopSpeed Modula-2:		
B-Tree Toolkit	149	135
Communications Toolkit	149	135
Compiler Kit	100	89
DOS 3-Pack	200	179
NETWORK PROGRAMMING		
Above LAN	495	395
Btrieve/N	595	459
Novell C Network Compiler	695	559
dBASE IV LAN Pack	995	645
FoxBASE +/ LAN	595	479
NetWare SQL	595	459
Paradox LAN Pack	995	697
Remote Procedure Calls	950	829

	LIST	OURS
OS/2 TOOLS		
Brief	199	155
Btrieve	595	449
CASE:PM	CALL	CALL
Epsilon	195	159
Greenleaf DataWindows	395	330
MKS LEX:YACC (OS/2)	399	339
MKS Toolkit (DOS & OS/2)	399	339
MS OS/2 Pres. Mgr. Softset	150	105
MS OS/2 Pres. Mgr. Toolkit	500	349
MultiScope	299	229
Panel Plus	495	395
PC-lint	139	101
PCYACC	395	359
Smalltalk/V PM	495	395
Vitamin C (OS/2)	225	165
XVT/PM	595	509
PASCAL LANGUAGE		
Asynch PLUS	149	115
B-tree Filer	125	99
MS QuickPASCAL	99	69
Object Professional	150	119
Power Tools PLUS/5.0	149	109
Topaz	75	67
Turbo Analyst	99	79
TurboMAGIC	199	179
Turbo Pascal 5.5	150	99
Turbo Pascal 5.5 Professional	250	169
Turbo-Plus 5.5	199	159
Turbo Professional 5.0	125	99
PROTOTYPING		
Dan Bricklin's Demo II	195	159
Instant Replay III	150	135
ProtoFinish	300	269
Show Partner F/X	350	319
Soft Demo	80	70
TRANSLATORS		
Bas_C Commercial	375	323
dBx Translator	550	467
FOR_C	575	519
PROMULA.FORTRAN	450	399
WINDOWS (MS) TOOLS		
Actor 2.0	699	559
Case:W	795	759
C-Talk/Views	450	375
dBFast/Windows	249	229
DialogCoder	499	479
MS Windows Development Kit	500	349
RFFlow	79	69
Whitewater Resource Toolkit	195	169
WinTrieve	395	339
ADDITIONAL LANGUAGES		
APL*PLUS	695	549
Janus Ada/Compiler System	300	269
Lattice RPG	1600	1469
Meridian AdaStudent	50	45
Meridian Ada Developer's Kit	1095	985
MKS AWK	99	85
Personal Rexx	150	139
Smalltalk-80 (386)	595	535
Smalltalk/V	100	85
Smalltalk/V 286	200	169

NEW RELEASES

SECOM *by Secure Communication Technologies*
SECOM is a secured encryption communications solution for PC's. SECOM supports IBM and IBM compatibles and gives you the ability to securely exchange mail, files and fax's while communicating bidirectionally. SECOM is DES approved and includes an Auto Session Key System.
List: $300 **Ours: $269**

Norton Backup *by Peter Norton Computing*
Speedy, reliable and easy to use. Reads the hard disk and writes to a floppy simultaneously. Norton Backup can restore from severely damaged disks. Saving and restoring can be done quickly by pointing and shooting through organized pop-up windows.
List: $149 **Ours: $99**

Actor 2.0 *by The Whitewater Group*
Create windows applications in significantly less time than it takes in C. Actor 2.0 adds three new features: ancestor binding, class variables and class initialization.
List: $699 **Ours: $559**

Service, Selection, $avings *(800) 445-7899*

Borland Spring Sale

	List	Ours
Paradox	725	479
Sidekick Plus	200	139
Turbo Assembler/Debugger	150	99
Turbo C 2.0	150	99
Turbo C Professional	250	159
Turbo Debugger & Tools	CALL	CALL
Turbo Pascal 5.5	150	99
Turbo Pascal Professional	250	169

APPLICATION SOFTWARE

	LIST	OURS
COMMUNICATIONS		
BLAST	250	225
Carbon Copy Plus	199	129
DeskLink	170	129
Laplink III	150	99
PC Anywhere III	145	99
Procomm Plus	75	50
SideTalk	120	90
DESKTOP PUBLISHING		
Adobe Illustrator	695	409
Corel Draw!	595	399
Gem Desktop Publisher	299	183
HALO DPE	195	139
Lattice HighStyle	375	319
MKS SQPS	495	479
PageMaker	795	509
Ventura Publisher	895	525
MATHEMATICS		
Derive	200	179
MathCAD	495	315
Mathematica 386	695	625
SCIENCE & ENGINEERING		
AutoCAD Release 10	3000	CALL
AutoSketch	150	95
ChiWriter	150	129
Control System Toolbox	495	375
CSS	495	469
DADiSP	895	759
Design CAD 3-D	400	292
Drafix Windows CAD	695	CALL
EXACT	475	380
Generic CADD Level 3	300	179
LABTECH Acquire	195	179
LABTECH Chrom	995	779
LABTECH Notebook	995	779
MICRO-CAP III	1495	1269
Orcad PCB	1495	CALL
PC TEX	249	229
SCHEMA II+	495	449
STATGRAPHICS	895	586
Tango CAD Pack	995	949
Tango PCB Series II	595	559
TECH*GRAPH*PAD	395	319
T^3	595	479
UTILITIES		
386 MAX	75	66
386 MAX Professional	129	115
above DISC	119	84
Command Plus	130	109
FASTBACK Plus	189	109
HeadRoom 2.0	130	109
Help Build	249	179
MACE GOLD	149	129
Magellan	139	99
Manifest 1.0	60	53
Memory Mate	70	47
MKS Toolkit	249	199
Move'em	89	79
Norton Commander	89	58
Norton Utilities	100	65
Norton Utilities Advanced	150	99
PC/Tools Deluxe	129	80
Pizazz Plus	149	79
QEMM/386	60	55
QRAM	80	69
SpinRite	89	69
XTreePro	129	109

PRODUCTS BY VENDOR

	LIST	OURS
DIGITALK		
Smalltalk/V	100	85
Communications	50	45
EGA/VGA Color Extension	50	45
Goodies #1, #2 or #3	50	45
Smalltalk/V 286	200	169
Smalltalk/V MAC	200	159
Smalltalk/V PM	495	395
IGC		
VM/386	245	199
VM/386 Multi-User	895	819
VM/386 Multi-User Starter	395	359
VM/386 NetPak	150	119
MICROSOFT		
MS BASIC Prof. Devel. System	495	349
MS C	495	349
MS COBOL	900	629
MS FORTRAN	450	299
MS Macro Assembler	150	105
MS OS/2 Present. Mgr. Toolkit	500	349
MS OS/2 Softset	150	105
MS Pascal	300	209
MS Programmer's Library	395	275
MS QuickBASIC 4.5	99	69
MS QuickC 2.0	99	69
MS QuickC w/ QuickAssembler	199	139
MS QuickPASCAL	99	69
MS Windows	99	69
MS Windows/386	195	139
MS Windows Development Kit	500	349
MS Word for Windows	495	349
MORTICE KERN SYSTEMS		
MKS AWK	99	79
MKS LEX:YACC	249	197
MKS MAKE	149	119
MKS Programming Platform	665	599
MKS RCS	189	149
MKS Software Mgmt. Team	299	239
MKS SQPS	495	479
MKS Toolkit	249	197
MKS Trilogy	119	105
MKS Vi	149	125
POCKET SOFT		
.RTLink	295	265
.RTLink PLUS	495	419
SAGE SOFTWARE/POLYTRON		
C Beautifier	50	42
Dan Bricklin's Demo II Program	199	159
PFinish	295	259
PFix86Plus	295	259
Plink86plus	495	395
PolyAWK	99	85
OS/2 Version	199	179
PolyBoost II	80	72
PolyDoc	199	169
PolyLibrarian	99	85
PolyLibrarian II	149	125
PolyMake 3.0	149	125
PolyShell	99	85
PolyXRef	99	85
Professional PVCS (Corporate)	495	419
w/ PolyMake for OS/2	695	589
PVCS/MAKE for OS/2	395	335
ZORTECH		
Zortech C Video Course	300	269
Zortech C++ Compiler V2.0	200	165
w/ source	300	259
Zortech C++ Debugger V2.0	150	129
Zortech C++ Developer's Edition	450	399
Zortech C++ Tools V2.0	150	129
Zortech C++ Video Course	500	449

Programmer's Policies

Phone Orders
Hours 9 AM-7 PM EST. We accept MasterCard, Visa, American Express. Include $4.00 per item for shipping and handling. All domestic shipments by UPS ground. Rush service available.

Mail Orders
POs by mail or fax are welcome. Please include phone number.

International Service
Phone number required with order. Call or fax for additional information.

Dealers and Corporate Accounts
Call for information.

Unbeatable Prices
We'll match nationally advertised prices. (Subject to same terms and conditions.)

Return Policy
30-day no-hassle return policy. Most manufacturer's products cannot be returned once disk seals are broken.

HELP/BUILD

An Automated Documentation System

HELP/BUILD is a complete help information and error screen generation tool. It allows you to develop any kind of pop-up help and error message system. You can create independent help systems which users call at the touch of a hot-key. You can link the help screens in any order to lead your users to the next information they should see. HELP/BUILD is menu driven and includes its own screen editor, optimizer, compiler, cross-reference generator and run module. Builds context sensitive help. Requires hard disk and 256K memory.

PACIFIC FIRMWARE MARKETING CORP.

List: $249
Ours: $179

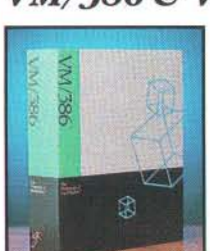
.RTLink/Plus

The only MS-DOS linker with CodeView support for multiple/nested overlays. Advanced technology allows more overlays, overlay areas, symbols, objects, classes, and sections than any other linker. Smart caching of most recently accessed overlay sections in conventional, expanded, and extended memory (new XMS specifications). Easy-to-use Profiler provides detailed performance statistics without source code modification. Run-Time Libraries (RTLs) speed development time and reduce disk space/modem-time requirements. Source code provided for overlay manager and Profiler's read-analysis utility. LIBLIST provides detailed information on inter-relationships between modules—great for optimizing and designing overlay schemes.

List: $495 Ours: $419

Pocket Soft Inc.

VM/386 & VM/386 NetPak



If you've ever had to wait for your computer to finish processing a task, you need VM/386. It lets you create lots of DOS sessions on your 386 PC, each running a different application on a full screen. All background sessions will continue to process whether you work in an application on your computer or go off to lunch. And with the addition of NetPak, each DOS session can interact with network files and peripherals. This is the only way you can multitask large programs without having to exit and re-enter the network.

IGC

Make VM/386 & NetPak your choices for bullet-proof multitasking and easy network access.

VM/386: $199
VM/386 NetPak: $119
Together: $299

The new
Programmer's Paradise®
catalogs have arrived!!

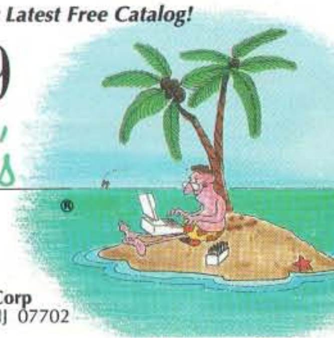
Call today for your FREE copy.

International: **201-389-9228**
Customer Service: **201-389-9229**
Fax: **201-389-9227**

New Corporate Phone #:
800-422-6507

Call or Write for Latest Free Catalog!

1-800-445-7899



Programmer's Paradise®

A Division of Voyager Software Corp
1163 Shrewsbury Ave., Shrewsbury, NJ 07702

(continued from page 8)

need not be modular, and a module may or may not be structured. Also, a module, program, or system that has "spaghetti code" has a structure. It is called a random structure.

One of the purposes of using modules is that the code is reusable. We use this all the time. The modules are in libraries. Examples are the Fortran library routine SIN(x), the Cobol COPYLIB file, and the C library routine sin(x). The proper term for module tree relationship is a "caller" module and a "called" module. This describes the relationship much better than the "father" – "son" model. Further, a module should not know anything about the "caller" module, other than what is in its argument list. Can you imagine the chaos that would result if we had to rewrite SIN(x) or sin(x) every time they had more than one caller?

Ned Logan
Seattle, Washington

Pascal Participation, Pleeez

Dear *DDJ*,

After reading Terry Ritter's letter entitled "Standardizing the Standardizing Process" in your February 1990 "Letters" column, I have the feeling that many readers may not understand the standardization process and will be given a false impression.

Membership on X3J9, the other X3 committees, and the IEEE committees is not restricted to some elitist group. Membership is open to all interested parties who are willing to participate. Users are especially encouraged to participate.

Decisions are not made in a back room behind closed doors. Committee meetings are open to the public with visitors and observers not only welcome but encouraged. Likewise, committee documents are open to the public and people who cannot attend meetings can become official observers and receive all committee mailings.

Consensus is the method by which most decisions are made both at the international level and at the domestic level for Pascal. However, it is the consensus of those who participate.

It was not only the consensus, but the unanimous vote of both the International Working Group on Pascal and the American National Standards Committee on Pascal that no action be taken on some of Mr. Ritter's comments for the Extended Pascal standard. The main reason for this was that major changes and development would have been required and it was felt that this should be handled separately rather than unduly delay the standard, which was in its final stages of review.

This does not mean that his comments have been shoved under the table. Many of the areas brought up by his comments, including exception handling, alphanumeric labels, and multiple arithmetic data types, are being worked on by the committee. The intent is to issue information bulletins, technical reports, and addenda to standards when work on them is completed. User participation is encouraged in this work, especially now when it is still in a nearly stage of development.

The committee is also now beginning to look at object-oriented extensions to Pascal, and has submitted a project proposal to its parent bodies for approval of this work. It is expected that approval will be received early this year. When this approval is received, announcements will be submitted to all publications (including *Dr. Dobb's* and similar user-oriented publications) that might have an interested audience.

People from Apple, Borland, Microsoft, and other vendors are planning to participating in the object work. User's views, and user participation, at this early stage would be especially welcome.

I encourage Mr. Ritter and other users to participate in Pascal and other standardization efforts. No one needs to elect you. All you need to do is participate.

For users that do not have financial support, there are organizations (such as SIGPLAN) that have funds allocated for this purpose.

To find out more about participating in the Pascal standards activity please contact me by letter, phone, FAX, or e-mail.

Thomas N. Turba
Chairman X3J9, Pascal
Unisys Corp., MS: WE3C
P.O. Box 64942
St. Paul, MN 55164
612-635-2349; 612-635-2003 (Fax)
NET: turba@rsvl.unisys.
comuunet!s5000!turba

There's More than One Way to Get From Pascal to C, or Return of the Living Fugu-Eaters

Dear *DDJ*,

I too enjoy Jeff Duntemann's writings, though not for the same reason as does Dale Lucas ("Letters," Jan. 1990). I love a good argument. So I get a kick out of reading Jeff's ravings against C, all the while thinking up incontrovertible (I'm sure they *must* be) refutations.

I do agree with him that C is ugly. It looks like Dagwood's dialogue after he hammers his thumb, %*!&( )*#$@! But I put up with that for the sake of the language's abilities. Jeff, on the other hand, sees no redeeming value in C, whatsoever, as we were so forcefully reminded in the January issue.

Recall that Dale Lucas asked him whether there's a way to call a third-party (*sans* source) C library's routines from a Turbo Pascal program. This little spark lit the fuse to one of Jeff's best tirades to date, in which he accused C programmers of acting macho, of neglecting to neck with their spouses and play with their dogs, and (this was the killer blow) of EATING FUGU!

Oh boy, did he give it to us. Unfortunately, he got so carried away with his ranting twaddle that he neglected to help his Pascal co-linguist. He told Dale to rewrite the whole library in Pascal!

If you don't mind taking advice from a fugu-eater, Dale, I think there's a way to hook those C routines. But first a question: Doesn't Turbo Pascal have something akin to Microsoft's "[C]" attribute, which you append to a procedure declaration to tell the compiler to use C's calling and naming conventions? Guess not, or the problem would be trivial and you wouldn't have written.

So you'll need to turn to a more powerful language — ummmh, let's say C — to write the hooks. Your third-party library will have given you a header file, for instance "WINDOW.H," declaring its functions. For example:

```
int WinCreate(int height, int width, int
                                  color);
void WinOpen(int winnumber, int
                       xcord, int ycord);
```

And so forth. Add to this file a new Pascal-callable hook function for each declaration, thusly:

```
int pascal HOOK_WinCreate(int height,
                  int width, int color) {
return( WinCreate(height, width, color
                                   )); }
void pascal HOOK_WinOpen(int win-
number, int xcord, int ycord.)
{WinOpen(winnumber, xcord, ycord);
                                 return;}
```

Rename the file, say to "HOOK.C," and compile it. Finally, translate the hook function prototypes into declarations for your Pascal modules:

```
function HOOK_WinCreate
(height,width,color : integer) : integer;
                                  extern;
procedure HOOK_WinOpen (winnum-
     ber, xcord,ycord : integer) :extern;
```

(Do I have that right? I read Pascal but speak it poorly.) The C code above works on my Watcom compiler, and

(continued on page 14)

FINALLY. A debugging tool tough enough to handle the DOS Nasties.

New Version 2.0

Nasty over-write? No sweat!

Soft-ICE memory range break points help you track down memory over-write problems whether you are doing the over-writing or another program is over-writing you.

Hung program? No problem!

When the system hangs, you now have hope. With Soft-ICE you can break out of hung programs no matter how bad the system has been trashed. And with Soft-ICE's back trace ranges you can re-play the instructions that led up to the crash.

Program too large? Not with Soft-ICE!

Soft-ICE runs entirely in extended memory. This means you can debug even the largest DOS programs. And since your program runs at the same address whether Soft-ICE is loaded or not you can find those subtle bugs that change when the starting address of your code changes.

System debugging? Soft-ICE is a natural!

Soft-ICE is ideal for full source level debugging of TSRs, interrupt service routines, self booting programs, DOS loadable device drivers, real-time kernels, non-DOS O/Ss and ROMs. Soft-ICE can even debug within DOS & BIOS.

How Soft-ICE Works

Soft-ICE uses the power of the 80386 to surround your program in a virtual machine. This gives you complete control of the DOS environment, while Soft-ICE runs safely in protected mode. Soft-ICE uses the 80386 to provide real-time break points on memory locations, memory ranges, execution, I/O ports, hardware & software interrupts. With Soft-ICE you get all the speed and power of a hardware-assisted debugger at a software price.

Don't want to switch debuggers?

You don't have to!

Soft-ICE can run stand-alone or it can add its powerful break points to the debugger you already use. Use your favorite debugger until you require Soft-ICE. Simply pop up the Soft-ICE window to set powerful real-time break points. When a break point is reached, your debugger will be activated automatically.

MagicCV with Soft-ICE

Using Soft-ICE with CodeView gives you the features necessary for professional level systems debugging. MagicCV and Soft-ICE can work in concert with CodeView to provide the most powerful debugging platform you will find anywhere.

"These may be the only two products I've seen in the last two or three years that exceeded my wildest expectations for power, compatibility and ease-of-use."
—Paul Mace
Paul Mace Software

Soft-ICE	$386
MagicCV	$199
MagicCV for Windows	$199
Buy Soft-ICE & MagicCV(W)	—Save $86.
Buy MagicCV and MagicCVW	—Save $100.
Buy All 3	—Save $186.

30 day money-back guarantee
Visa, MasterCard and
AmEx accepted

1988 BYTE Award of Distinction

New Product / New Idea

- Find overwrites and un-initialized pointers automatically
- All the protection of a protected O/S under DOS

Bounds Checker - $249.

Nu-Mega TECHNOLOGIES

CALL TODAY (603) 888-2386
or FAX (603) 888-2465

RUN CODEVIEW IN 8K

MagicCV

CodeView is a great integrated debugger, but it uses over 200K of conventional memory. MagicCV uses advanced features of the 80386 to load CodeView and symbols in extended memory. This allows MagicCV to run CodeView in less than 8K of conventional memory on your 80386 PC.

NEW—Version 2.0 includes EMS 4.0 driver.
Attention Windows Developers!
Version available for CVW.

P.O. BOX 7607 ■ NASHUA, NH ■ 03060-7607

CIRCLE NO. 174 ON READER SERVICE CARD

(continued from page 12)
ought to work on QuickC as well. Of course, I've begged the more difficult questions like memory models and translating C strings and structures into Pascal. But this might be enough to get you started.

A couple of closing questions. Dale . . . wouldn't it be easier just to code your app in a real-man's language like C? And Jeff . . . what the hell IS fugu, anyhow?

Bob Twilling
Bozeman, Montana

Dear *DDJ*,
In your January issue, you printed a letter from Dale Lucas asking for help interfacing Turbo Pascal to C. Jeff Duntemann's response spent more effort bashing C than helping Mr. Lucas. I'm not particularly fond of C either, but I think I have a very simple solution.

I know nothing about Turbo Pascal, but if it uses (or can be made to use) the same calling convention as Microsoft Pascal, we're in luck. Simply use Microsoft QuickC to create one-line helper functions that translate the calling conventions. These helper functions would be declared as "pascal" functions, and thus be callable directly by Pascal. The only statement in the function would be a call to the library function using the C calling convention. For example:

```
int cdecl foobarC(int,int); /* This is in
                                the library */
int pascal foobarPascal(int arg1, int arg2)
{
return ( foobarC(arg1, arg2) );
}
```

This assumes the C-based library has a function called *foobarC( )*, which has two integer arguments and an integer return value. The function *foobarPascal( )* passes the arguments in the return value out.

Tim Paterson
Renton, Washington

Dear *DDJ*,
This letter is in response to Jeff Duntemann's answer to Dale Lucas's letter in the January 1990 issue of *Dr. Dobb's Journal*.

I disagree with Jeff's answer. A Pascal routine can call a C routine by using an impedance matching routine written in assembly. The routine takes the Pascal arguments, pushes them on the stack, calls the C routine, cleans up the arguments pushed, and then cleans up the stack for the Pascal caller. A macro can be built, which has the Pascal entry point, the matching C entry point, and the size of the arguments in bytes. The macro *CHook* in Listing One (below) implements this.

The Pascal programmer simply calls the Pascal entry point. The impedance matcher handles the language differences and returns. Simple, easy, and direct. Much better than recoding a debugged, commercial library.

By placing the code in a macro, the user can just build a table of macro calls which reflect all entries to the C library. Each macro expands and builds the impedance matching code for each library entry point.

Some notes involving the use of the macro:

- The argument size is given in bytes, not number of arguments. You must determine the number of bytes by adding the number of bytes in each argument that is passed.
- This impedance matcher will work for Pascal procedures. For functions, you will have to make sure that your flavor of Pascal uses AX for 16-bit return values and DX:AX for 32-bit return values. If you need to map the function return values, just add this to the macro.
- I coded this macro for large model programs. Small model programs must adjust the value added to SI from 6 to 4.
- If SS matches DS at all times, the push/load/pop of DS can be removed.
- The argument transfer can be sped up by using a MOVSW and dividing the count stored in CX by 2. This should always work because C requires the minimum argument by an int (2 bytes under MSC).
- If the size of the arguments is 0, the argument transfer code can be eliminated using conditional assembly.
- Normally, the name of a C routine starts with an underscore. This could be included in the macro instead of requiring an underscore for every *CHook* invocation.
- The impedance match does take time. If you have a very time-critical call, you may have to recode the routine in Pascal or directly in assembly. However, the macro can get you up and running quickly.

This code has not been tested with Turbo Pascal. It has only been tested using a Microsoft C program (Listing One) that calls the Pascal entry using the pascal keyword. If the macro doesn't work with Turbo Pascal, it should only take a small amount of tweaking to make it work. The key is to draw a picture of your stack frame and test it in Debug, following the argument flow.

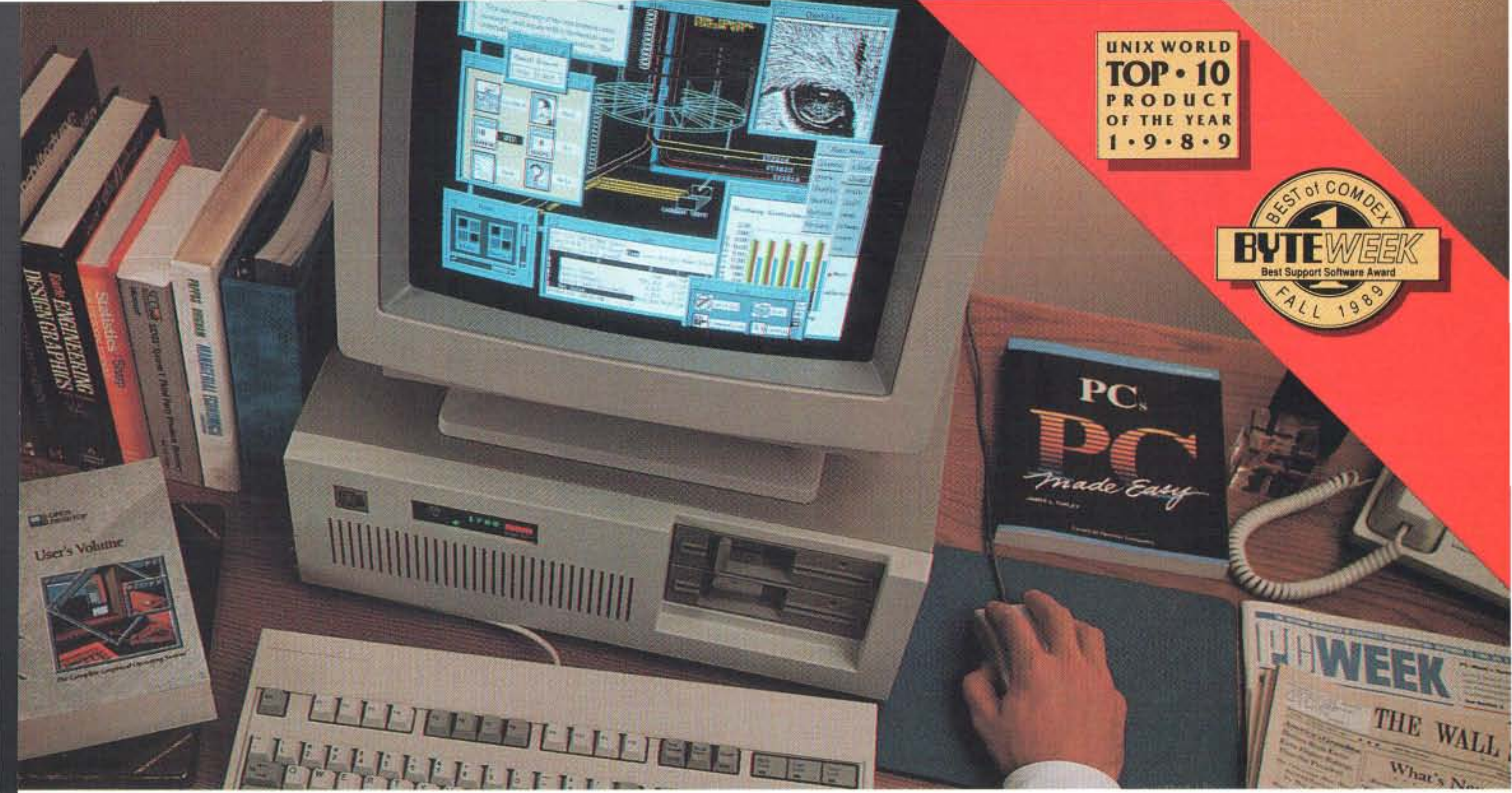
Jim Shimandle, Primary Syncretics
Santa Clara, Calif.

DDJ

Listing One

```
;
; c2pas.asm
; C/PASCAL impedence matching module
;
        .model  large
code    segment para public 'code'

;-----------------------------------------------------------------------

CHook   MACRO   PascalEntry, CEntry, ArgSize
        EXTRN   CEntry:FAR
        PUBLIC  PascalEntry
PascalEntry     PROC

        PUSH    BP                      ; Make stack frame
        MOV     BP, SP

        PUSH    CX                      ; Save registers
        PUSH    SI
        PUSH    DI
        PUSH    DS

        MOV     CX, SS                  ; Set DS to point to stack
        MOV     DS, CX

        SUB     SP, ArgSize             ; Save space for arguments
        MOV     CX, ArgSize             ; Set count for arg transfer
        MOV     SI, BP                  ; Get frame pointer
        ADD     SI, 6                   ; Point to start of PASCAL arguments
        MOV     DI, SP                  ; Point to start of C arguments
        CLD                             ; Move is up
        REP MOVSB                       ; Move the arguments

        CALL    CEntry                  ; Call the C routine
        ADD     SP, ArgSize             ; Remove arguments from stack

        POP     DS                      ; Restore registers
        POP     DI
        POP     SI
        POP     CX

        POP     BP                      ; Restore frame
        RET     ArgSize                 ; Exit

PascalEntry     ENDP
        ENDM

;-----------------------------------------------------------------------

; Invoke macro for test routine

CHook   p_sum3, _c_sum3, 6              ; Invoke macro for:
                                        ;   p_sum3 is PASCAL call
                                        ;   c_sum3 is C library routine
                                        ;   6 is the number of argument bytes

;-----------------------------------------------------------------------

code    ends
        end
;
; end of c2pas.asm
;
```

End Listing



INSTANT WORKSTATION. JUST ADD OPEN DESKTOP.

Take a look at the vast majority of graphical workstations developed over the past decade and you'll see something they all have in common:

An integrated UNIX® System environment.

Now take a look at the vast majority of businesses that have put computing power directly onto their office desktops over the past decade, and you'll see something they all have in common:

Industry-standard personal computers.

It doesn't take a computer to forecast the platform that's going to put graphical workstations on the vast majority of business and engineering desktops in the <u>next</u> decade:

An integrated UNIX System environment for industry-standard personal computers.

And that's what Open Desktop™ is all about.

Open Desktop is the complete graphical operating system that's built on the most popular UNIX System platform of all time—SCO™. And it lets you create your own networked, icon-driven workstation environment using the industry-standard 386 or 486 computers and peripherals of your choice.

In a single, easy-to-use, fully supported—and completely integrated—package, Open Desktop delivers:

- the full 32-bit, multitasking computing power of SCO UNIX System V/386
- compliance with POSIX™ and X/Open® standards
- an OSF/Motif™-based, Presentation Manager-compatible, graphical user interface
- distributed SQL database management services
- compatibility with existing DOS, XENIX®, and UNIX System applications and data files
- NFS™, TCP/IP, and LAN Manager networking facilities

And all at an unbelievably affordable price.

Discover the complete graphical operating system that leading companies worldwide are choosing as their development platform for the '90s—and using to turn their 386 and 486 PCs into instant workstations today.

Open Desktop from SCO.

OPEN DESKTOP.
The Complete Graphical Operating System

SCO™
THE SANTA CRUZ OPERATION

For more information, call SCO today and ask for ext. 8401

CIRCLE NO. 42 ON READER SERVICE CARD

(800) SCO-UNIX (726-8649) or (408)425-7222 FAX: (408)458-4227 E-MAIL:...!uunet!sco!info info@sco.COM

Bidirectional Associative Memory Systems in C++

Recent innovation makes associative memory practical for real-world problems

Adam Blum

Content-addressability was always a goal of early neural network pioneers. It is a quest that has been pursued by computer scientists in general for decades. However, the goal has proved highly elusive. Search time has always depended on the amount of data stored, although much research has gone into reducing the slope of this curve. Real-time pattern recognition (as applied to any number of fields, be it speech recognition, radar signature identification, or part classification) is still far from reality. One particular neural-network construct, bidirectional associative memory (or BAM), has promised some solution to this problem.

I'll first describe the BAM concept, then show you how a relatively recent construct, the Bam System, can make it immediately feasible for real problems. Finally, I'll present an actual implementation of the Bam System written in C++.

As developed by Bart Kosko, BAMs are a neural-network-based attempt at content-addressable memories. They are based on a two-layer feedback neural network. They attempt to encode m pattern pairs (A_i,B_i) where $A_i \varepsilon \{-1,+1\}^n$ and $B_i \varepsilon \{-1,+1\}^p$ in an $n \times p$ matrix M. BAMs are globally stable and provide instant recall of either of the two-pattern pair elements. However, BAMs face some limitations. For large pattern lengths, n, storage requirements increase $O(n^2)$. More importantly, storage capacity is only, on an average, $m < \min(n,p)$. Thus, for moderate pattern lengths, capacity of the matrix M becomes a problem. Recent research promises help for this problem. However, some initial description of BAMs should be made.

Adam is a programmer analyst at Ketron Inc. of Arlington, Virginia, and is the principal developer of several commercial software packages. His interests include compiler design, C++, and (of course) applications of neural nets. He can be contacted at 1700 N. Moore St., Ste. 1710, Arlington, VA 22209, or on CompuServe at 72650,1773.

Encoding

BAM encoding is accomplished by simply summing the correlation matrices of each of the pattern pairs. That is, the matrix that encodes the first m pattern pairs, M, is simply:

$$M = \sum_{i=1}^{m} A_i^T B_i$$

Thus, to encode a pattern pair, simply produce its correlation matrix, $A_i^T B_i$, and add the values to the current matrix M. For discrete implementations, it so happens that the matrix arithmetic works out better if 0s and 1s are encoded as -1s and +1s. So the first step in the process will be to convert any {0,1} string to {-1,+1}. Example 1 shows this process.

Note that we can erase association (A_i,B_i) from M by adding $-X_iTY_i$ to M. But if we are using a {-1,+1} representation, this is the same as adding (A_i,B_iC) or (A_iC,B_i) to M (where C represents the pattern's complement). This fact will become important in our implementation of the BAM system.

Decoding

After we have "trained" our BAM with the m pattern pairs (A_i,B_i), we want the BAM to recall pattern B_i every time A_i is presented to the matrix (and, conversely, recall A_i every time B_i is presented to the matrix). It turns out that BAMs also have the property that B_i will be recalled every time something close to A_i is presented. Example 2 outlines the steps involved in the decoding process.

But it won't go on forever. As shown in Example 2, eventually the fields will "resonate" to steady patterns. This property of BAMs is called "global stability." Lyapunov energy functions allow us to prove that BAMs are globally stable.

Energy Functions and Stability

Lyapunov showed that any function expressed in terms of the system parameters that is zero at the origin and has nonincreasing changes is globally stable.

An energy function for the BAM can be $E(A,B) = -AWB^T$. This function is obviously zero at the origin (that is, zero when A and B are zero). We just need to show that it has nonincreasing changes. Well, $\Delta E_A(A,B) = -AWB^T$ and by the definition of our function *f*, each A_i in A will be positive only if W_iB is positive. If A_i is negative, W_iB must also be negative. Thus the change in energy will always be negative or zero. The system is thus globally stable.

Adaptive BAM

As we have just described it, the connection matrix M is simply the sum of the correlation matrices of the patterns presented to it. We can use more sophisticated equations to allow faster convergence or more accurate recall. As long as such equations can also be shown to converge, we should have no problem with this.

The simplest of these learning laws is called Hebb's law: $m_{ij} = -m_{ij} + f_i(x_i) * f_j(y_j)$, where m_{ij} is the connection weight between the neuron x_i and neuron y_j, and f_i and f_j are the threshold activation functions for x and y, respectively.

Other laws that could be used include competitive learning and differential Hebb; there is much research on which of these is most effective. In our implementation, we will be presenting a simple nonadaptive BAM. However, it is easily extensible to the learning function of choice.

Problems

BAM faces two problems, the first of which is that the amount of storage taken up varies $O(n^2)$, where n is the pattern length (actually, it will vary $O(np)$ where n is pattern length of A and p is pattern length of B).

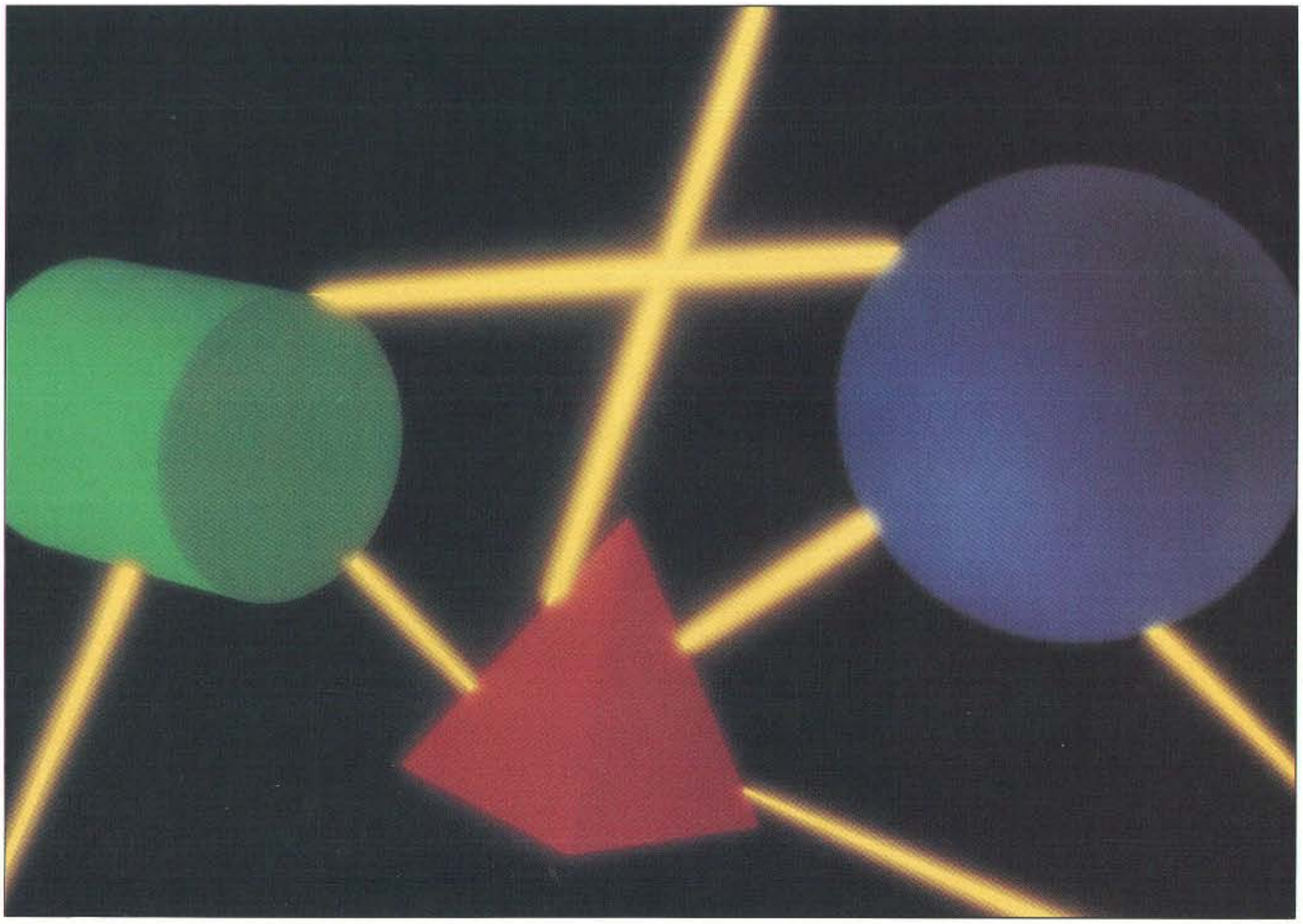
The second problem — capacity — is more critical. Reliable retrieval of associations begins to degrade when the number of patterns stored, m, is greater than the minimum of the two-pattern dimensions. In other words, to be reliable the matrix capacity is $m < \min(n,p)$.

For large pattern lengths, this is not so much of a problem, but many applications have inherently moderate pattern lengths. We intuitively find it almost obvious that if a BAM can store only up to the minimum of its pattern lengths, it will be virtually useless for real-world applications.

BAM Systems

In 1989, Patrick Simpson of General Dynamics published a paper introducing the concept of a "BAM System." This is a rather uninformative name for a system that allows for multiple matrices when one matrix's capacity is saturated. Perhaps a better name would be "Multi-Matrix BAM" or, because each matrix is just a representation of the connectivity between the two patterns, "Multi-Connective BAM." Anyway, it is an inventive way to overcome the severe problem of matrix capacity.

The Bam System operates as follows: Pattern pairs are encoded one by one in a single BAM matrix, M_1. After each pattern pair is encoded, the matrix must be tested to ensure that each pattern pair stored can be recalled. If a pattern pair cannot be recalled, the current pair is removed from the matrix. We then attempt to store the pair in another connection matrix. We continue to try to store it in other matrices, M_i, until it is stored such that all pattern pairs in that matrix



Menuet.

SIMPLE.
AND SIMPLY THE
MOST POWERFUL
GUI AVAILABLE.

Object oriented, extensible, sophisticated, but most of all, intuitively simple. Menuet is the most thoroughly designed and easy to use Graphical User Interface (GUI) for the DOS environment.

A complete development environment providing over 350 callable functions, Menuet supports virtually all GUI constructs including pull-down menus, dialog boxes, scrollable lists, data entry forms, icon and button panels, and much, much more.

ACTUAL SCREEN DISPLAY SHOWING MENUET'S DISTINCTIVE AESTHETICS AND MULTIPLE, OVERLAPPING WINDOWING CAPABILITIES.

Menuet also offers the powerful **Interface Design Tool**™ which allows you to interactively edit and prototype your applications with automatic source code generation.

Menuet. **$325** No Royalties. Source code available, call for pricing.

Available for Turbo C, Microsoft C, Zortech C, and C++. Requires MetaWINDOW.

CIRCLE NO. 286 ON READER SERVICE CARD

Call for your free demo.
303.494.8865

ITHACA STREET SOFTWARE

1145 ITHACA DRIVE, BOULDER, CO. 80303, USA

can be recalled successfully. The pattern association is then permanently stored in this matrix.

Decoding, that is presenting one-half of a pattern and recalling the other half of the pair, is a bit more complicated. Because we now have several matrices storing pattern associations, we don't know which one is the correct one to look in to recall the pattern pair. To choose which pattern pair to recall from each matrix, we use the following criterion.

We determine all the returned pattern pairs (X_i, Y_i) that

If we are trying to encode

$A_1 = (101010)\ B_1 = (1100)$
$A_2 = (111000)\ B_2 = (1010)$

we first convert to {-1,+1}.

$X_1 = (1\ -1\ 1\ -1\ 1\ -1)$ $\qquad Y_1 = (1\ 1\ -1\ -1)$
$X_2 = (1\ 1\ 1\ -1\ -1\ -1)$ $\qquad Y_2 = (1\ -1\ 1\ -1)$

$$X_1^T Y_1 = \begin{array}{rrrr} 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ -1 & -1 & 1 & 1 \end{array}$$

$$X_1^T Y_2 = \begin{array}{rrrr} 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{array}$$

$$M = \begin{array}{rrrr} 2 & 0 & 0 & -2 \\ 0 & -2 & 2 & 0 \\ 2 & 0 & 0 & -2 \\ -2 & 2 & 0 & 2 \\ 0 & 2 & -2 & 0 \\ -2 & 0 & 0 & 2 \end{array}$$

Example 1: *The encoding process*

Each neuron b_i in field Fb (Fa and Fb will be used to refer to the two pattern fields A and B) receives a gated input of all the neurons in Fa with a nonlinear threshold function applied. In our bipolar discrete example a typical function might be:

$$\begin{array}{l} f(x,y) = 1 \text{ if } x > 0 \\ y \text{ if } x = 0 \\ 0 \text{ if } x < 0 \end{array}$$

We now have a pattern B_1. However, we aren't done yet. The output from pattern B is then fed back through the transpose of matrix M to produce pattern A_1. That is, each neuron A_i in A receives gated input from each neuron B_j in B and applies the same threshold function to it.

A_1 is then sent back through the matrix again to produce B_2, and on this goes.

$$\begin{array}{l} A \dashrightarrow F(AM) \dashrightarrow B_1 \\ A_1 \leftarrow F(B_1 M^T) \leftarrow B_1 \\ A_1 \dashrightarrow F(A_1 M) \dashrightarrow B_2 \\ \quad\cdot \\ \quad\cdot \\ A_i \dashrightarrow F(A_i) \dashrightarrow B_i \\ A_i \dashrightarrow F(B_i M) \dashrightarrow B_i \end{array}$$

Example 2: *The decoding process*

NeuralWare, Inc.

World's Leading Supplier of Neural Network Development Systems

presents

THE HOTTEST PRODUCTS

UNDER THE SUN!

NeuralWorks Professional II and Designer Pack

To: Researchers and Developers

Subject: Short Quiz

How do you make your newly developed neural network part of your product?

☐ 1) Develop a network with a neural network simulator and write your own code to interpret and execute the results.

☐ 2) Start from scratch using a "Neural Network Language" and write your own development environment.

☐ 3) Roll your own network and development environment from scratch.

☐ 4) None of the above!

The answer is "4" - none of the above. NeuralWorks Professional II in conjunction with Designer Pack makes all other methods of neural development obsolete. The development environment (NeuralWorks Professional II) is seamlessly integrated with the deployment environment (NeuralWorks Designer Pack). It is easy to move back and forth between the two of them. You can try out as many networks and variations as you want. And when you're ready, you have a "C" program with a well-defined interface which will run identically to the network you proto-typed!

NeuralWorks Professional II with Designer Pack . . . . . *The logical alternative*

Together with NeuralWorks Professional II, Designer Pack is the most elegant method for designing, testing and deploying neural networks. The development blends naturally with application production. There is no need to learn special "Neural Network Programming Languages" which may be restricted to one or two special hardware systems. Build a network in a way that makes sense: graphically. Prototype your interfaces with User I/O. Turn it all into useable application code with Designer Pack. Nothing wasted. No re-design. No re-learning. Simple. Elegant. Seamless.

The sensible approach to neural network development.

NeuralWorks Professional II and NeuralWorks Designer Pack are available on the **Sun 3, 4, 386i, N-Cube, MacIntosh, PC AT, XT, PS2** and compatible machines.

Call TODAY or write for information about NeuralWare's software, seminars, and custom engineering services. Ask for Jane Klimasauskas, Vice-President Sales and Marketing.

Authorized Distributors:

Electronic Associates GmbH
FranzstraBe 107
D-5100 Aachen
West Germany
Telephone: 011-49(241)26041/42

Scientific Computers Limited
50 Victoria Road
Burgess Hill
West Sussex RH15 9LW
England
Telephone: 011-44(444)65101

NovaCast Expert Systems
Soft Center
S-37200 Ronneby, Sweden
Telephone: 011-46(457)718-30

Informagic, S.A.
Avgda, de Roma, 152
Barcelona 08011
Spain
Telephone: 011-34(3)254-6235

Electronic Associates Sarl
25-27 Rue Ginoux
75737 Paris Cedex 15
France
Telephone: 011-33(1)45770813

Nichimen Data System Corp.
Kyodo Building
3-2, 1-Chome Nihonbashi-honcho
Chuo-ku, Tokyo 103, Japan
Telephone: 011-81(3)241-2611

NeuralWare
INCORPORATED

Penn Center West-Bldg. IV, Suite 227
Pittsburgh, PA 15276
Phone: 412-787-8222
Fax: 412-787-8220

(continued from page 18)

have the same energy as the pair (A,Y_i) (where A is the presented pattern). Of these patterns we choose that pattern pair whose energy is closest to the matrix's orthogonal BAM energy. (Orthogonal BAM energy is the energy a matrix would have if all its stored patterns were orthogonal, which turns out to be equal to the negative of the product of the pattern lengths, $E^* = -np$. Energy of a pattern pair can be calculated the same way as in our previous discussions, $E = -XMY^T$, where X and Y are the two patterns.)

There are some problems with the Bam System. In order to keep checking that the patterns were stored reliably in each matrix (without corrupting the other patterns already in the matrix) the patterns need to be stored separately. Also, the need to compute the "best" recall from each of the BAM matrices could be computationally prohibitive. Parallel hardware (which, presumably, a BAM would be running on anyway) could possibly ease this burden.

The Implementation

C++ provides an excellent tool for implementing neural nets in general and BAMs in particular. Most of the constructs in this discussion of BAMs were vectors and matrices. This is a classic application of object-oriented programming. Classes for vectors and matrices should go a long way toward making the implementation easier. Listing One (page 84) is BAM.HPP, the BAM header file that contains the class definitions. Listing Two (page 84) is BAM.CPP, the BAM program file that contains the BAM implementation.

The vector class is implemented in classic fashion (almost identical to Stroustrup's). Methods are provided for assignment, multiplication by scalar constant, and dot product. This is all that is really necessary, but a few more methods are provided for completeness. Streams input and output are provided to read the patterns in and display patterns to the user. The streams functions do the necessary (0,1) to (-1, +1) conversion discussed earlier.

The matrix class is implemented as an array of pointers (int **), with indicators of the number of rows and columns. It could conceivably have been implemented as an array of vector objects. I chose representation for efficiency. There are several constructors provided. The first simply initializes the matrix from specified dimensions. These dimensions default to the particular application's two pattern lengths (specified by the ROWS and COLS constants). Other constructors are provided to form a matrix from a pair of vectors by multiplying one vector by the transpose of another ($M=AB^T$). Standard matrix arithmetic functions are included. Methods are also provided to form a vector from a row or column "slice" of the matrix. Streams output is provided for debugging diagnostics.

Another fundamental construct is the pattern pair. This is, after all, what the BAM lets us do — retrieve pattern associations. Pattern pairs are represented by the "vecpair" (vector pair) class. An "encode" operation will encode a vecpair. A "recall" operation will return a vecpair, when supplied with a pattern (or "vec").

Once we have these vector, matrix, and vector pair classes, implementing the BAM is fairly simple. The BAM is essentially just a matrix. We use the C++ inheritance mechanism to inherit the matrix and all its functions. We made the matrix's data structures "protected" instead of "private" so the derived BAM matrix class could use the matrix's data structures. We now just add a vecpair pointer for the pattern pair list and the BAM matrix functions.

(continued on page 24)

Tired of Fishing in a Sea of Software Products? We Can Help You Catch "The Big One."

EASE OF USE: Simple to use CUA (Common User Access) menus are easy to create, even for beginners, using the menu design toolkit included with WorkStation Manager™.

POWER: Develop applications ranging from simple menus that execute purchased software, to complex, intelligent applications that interact with mainframes. Scripts that completely automate host operations are simple to create and use. WorkStation Manager/API supports standard 3270 SNA products with HLLAPI or low level API interfaces, providing consistent script writing in all cases.

FLEXIBILITY: WorkStation Manager is available in a number of versions to suit any PC environment. From stand alone workstations, to network or workgroup workstations, to IBM PC's running with 3270 terminal emulators, WorkStation Manager can be quickly customized to your requirements.

COST: WorkStation Manager is priced at $195 for a single user version and less than $80.00 a workstation for a 5 user LAN version. Quantity discounts and corporate licensing are also available.

SUPPORT: Our technical support team is always available to answer your questions or offer professional advice. A warranty registration program is included. Also available is a Maintenance Enhancement Plan.

DON'T DELAY, call for more information and a demonstration disk. *Don't let WorkStation Manager be "The Big One" that got away.*

McCarty Associates, Inc.
Software Development • Consulting

Telephone: 203-388-6994
FAX: 203-388-6826

CIRCLE NO. 86 ON READER SERVICE CARD

System Architect is perfect for the beginning CASE user, yet has the power and flexibility to meet the needs of the most experienced users and largest applications.

System Architect has the power and flexibility you need from a CASE product. It works with today's most popular methodologies, including DeMarco/Yourdon, Gane & Sarson, Ward & Mellor (real-time), entity relation diagrams, decomposition diagrams, object oriented design (optional), state transition diagrams, and flow charts. It supports an integrated data dictionary/encyclopedia, and allows multi-user support both with and without a network. What's more, System Architect's open architecture lets you easily import and export data to other products.

"We're surprised with its flexibility and much taken with the idea of being able to link different kinds of diagrams, in effect moving between analysis and design and back again..." Cutter Information's CASE Strategies, July '89

Yet System Architect is more than just powerful. It's easy to use. Microsoft Windows-based and graphics oriented, System Architect lets you get up and running right away. And if you get lost, you can call on a context-sensitive help facility as well as a novice mode.

"We found System Architect to be extremely easy to use." ISI Systems, CASE Trends, Nov/Dec '89

At **$1,395,** System Architect is quite affordable. It runs on almost any PC, and it won't run away with your budget.

"Think productivity has to be pricey? Think again. This product is truly a price performance leader" System Builder, Oct/Nov '89

If you're looking for a CASE product with power, ease of use, and affordability, look to System Architect. It's a concept whose time has come.

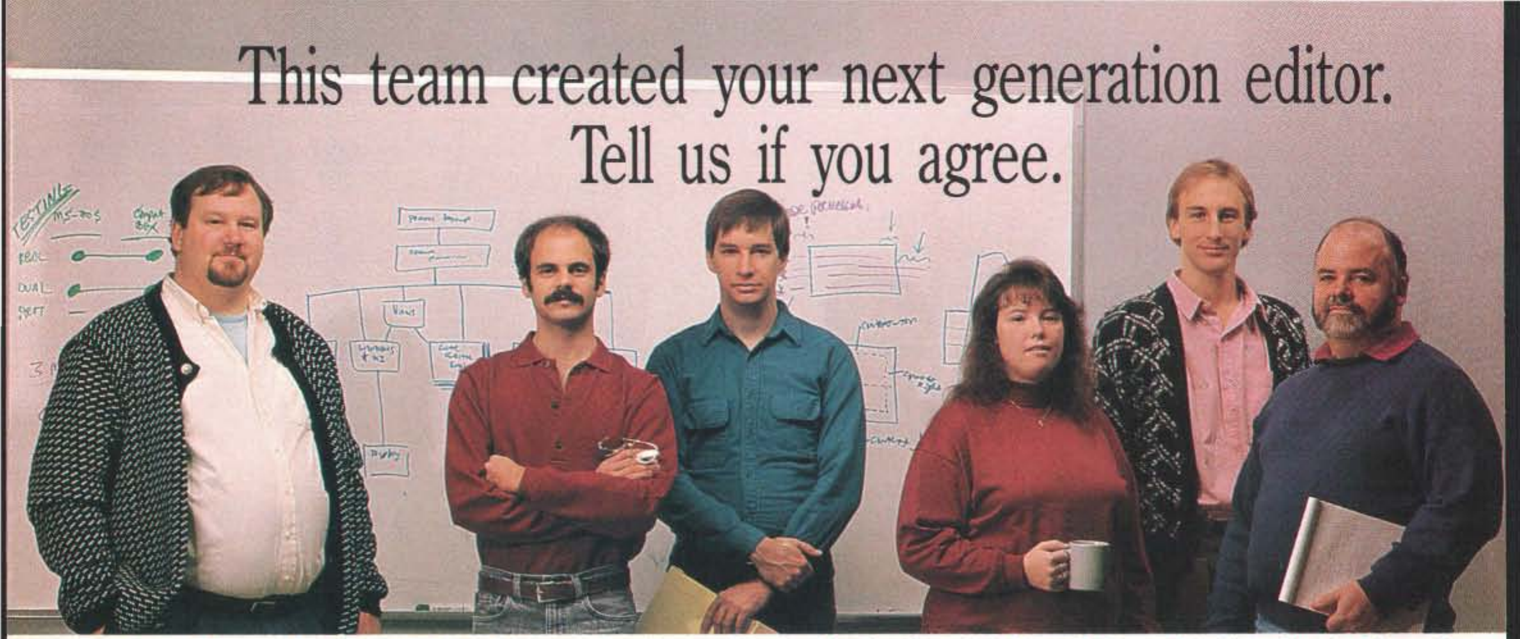
System**Architect** ™

Popkin Software & Systems Incorporated
11 Park Place, 19th Floor, NY, NY 10007
TO ORDER, CALL (212) 571-3434
Fax: (212) 571-3436

Supporting IBM's AD/Cycle

CIRCLE NO. 321 ON READER SERVICE CARD

This team created your next generation editor. Tell us if you agree.



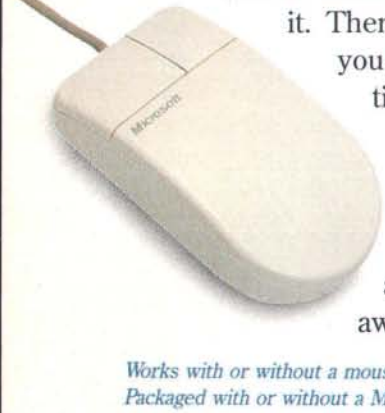
From left to right: ***Jim Besemer****, Project Manager, Specification & Design, User Interface;* ***Skip Russell****, System Primitives, Function Library, Error Recovery, Operating System Interface, Configuration and Setup;* ***Pat Thompson****, Extension Language/Compiler and Debugger;* ***Robynne Major-Jones****,* ***Howard Mercier*** *(not shown) Quality Assurance and Testing;* ***Eric Johnson****, Editor Core, Help Facility;* ***Milo Wadlin****, User Documentation.*

We talk to developers every day. New development environments and the size and complexity of projects are making their 80's editors obsolete. They told us they need a new editor for the 90's. So for two years this team worked in secret to create it.

Announcing the Sage Professional Editor

We believe they succeeded magnificently. Now we'd like to know what you think.

Try the Sage Professional Editor for thirty days with no risk. Pound on it. Compare it to any editor you've ever used. Then call our team at 1-800-547-4000 and tell them how you like it. There's no risk to you, no obligation. Not much risk for us either. We won't be able to pry it away from you.

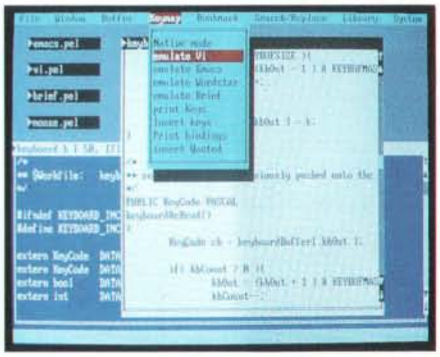


Works with or without a mouse. Packaged with or without a Microsoft® Mouse.

Right out of the box you'll be more productive with our editor than any you use today. The instant installation, elegant mouse support, advanced user interface, and point-and-shoot help get you running immediately. If you prefer the commands and keystrokes of a popular editor, our turnkey emulations duplicate them precisely, and you still gain the Sage Professional Editor's advanced features, windowing capabilities and powerful engine.

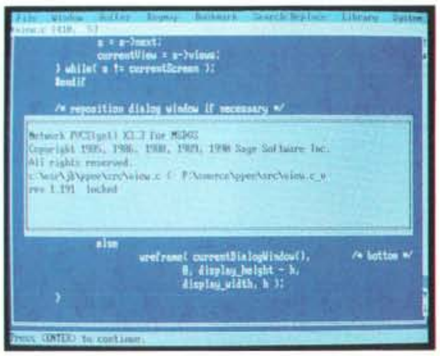
Later you'll make this editor truly yours by configuring the interface as you prefer. Every feature can be turned on or off as you like - from a clean screen, to tiled windows, to overlapping windows in various colors, pull-down menus, rulers, scroll bars, and line numbers - choose any or all and place them as you like.

Use the editor with or without a mouse - all functions are available without lifting your fingers from the keyboard. But the click-select windows, scroll bars, zoom, shrink-to-icon, block text selects, and other speedy mouse actions make editing extremely efficient.



```
#ifndef KEYBOARD_INC
#define KEYBOARD_INC

extern KeyCode   DATA
extern KeyCode   DATA
extern bool      DATA
extern int       DATA
```

Make our interface what you prefer, from clean screen to multi-window with drop-down menus and icons.



```
                s = s->next;
                currentView = s->views;
        } while( s != currentScreen );
        #endif

        /* reposition dialog window if necessary */

Network PVCS(get) X1.3 for MS-DOS
Copyright 1985, 1986, 1988, 1989, 1990 Sage Software Inc.
All rights reserved.
rev 1.191  locked

                else
                        wrefrane( currentDialogWindow(),       /* bottom */
                                0, display_height - h,
                                display_width, h );
        }
Press <ENTER> to continue.
```

The editor environment provides seamless integration to the Polytron Version Control System (PVCS) or any other tool you care to connect.

Sage Software, Inc., 1700 NW 167th Place, Beaverton, OR 97006, (503) 645-1150, Fax (503) 645-4576, Int'l. (301) 230-3307, Int'l Fax (301) 984-3047.

Pop open the DOS window and the editor shrinks to just 4k. So you can back-task to compilers and other tools without leaving the editor.

This package is stuffed with value. It includes MS-DOS, OS/2 and Dual Mode versions on both 5 1/4" and 3 1/2" disks, templates for popular languages, and you can buy it with or without a bundled Microsoft® Mouse.

But the reason you will become addicted to the Sage Professional Editor is the power behind that pretty face. We looked under the hood of other editors and found wimpy editing engines dressed up with all sorts of bells and whistles. Unacceptable. We wanted an industrial strength engine. So we started with a powerhouse virtual memory system that allows us to edit huge files (up to 100MEG) in as many as 256 windows - over two billion lines. It makes maximum

TurnKey Emulation Of

Vi | Brief | Wordstar

Epsilon/EMACS | Others

use of available memory (including EMS) and uses advanced processes to safely minimize file I/O. All higher level services use this powerful VM scheme. Consequently, there are no size constraints on the macro library and no limit to Undo/Redo. You can have 1000 bookmarks, anchors and saved positions per buffer.

Sage Software, Inc. has acquired Polytron, the company that brought you PVCS and PolyMake. Now this group proudly introduces their Professional Editor under the Sage banner.

DEVELOPMENT POLYTRON TOOLS

"When I saw how well my mouse works with this new editor I had to try it. I became a convert when I saw the power of the extension language. This editor can be whatever I need it to be. I couldn't give it up."

Terry Ham,
ImageBuilder Software

And then there's the extension language. The Sage Professional Editor uses a C-like extension language (100 percent PolyAWK compatible) and compiler/debugger that programmers find immediately intuitive. The function library gives access to nearly everything. From mouse click to menu, from word processing to external hooks. You can build the environment you need or want with the editor as the front end to your favorite tools. The seamless integration of the Polytron Version Control System (PVCS) is a sterling example of how cleanly you can hook external programs.

Emulations of Vi, Brief, EMACS, and WordStar, were written with the extension language. The source code for each is included. If you are addicted to some obscure editor you can modify a copy of emulator source and quickly make your own. Source for the menuing system, mouse interface, help and many other services is also included, so you can deeply modify the editor to be exactly what you want. Brief users who have a library of macros can use our translator to recompile them into our more useful and powerful C-like extension language.

CIRCLE NO. 190 ON READER SERVICE CARD

It's easy to make the Sage Professional Editor a standard for your team or even the entire company:

1. Turn-key emulation means every programmer can use it immediately. Macros written to aid users apply to all emulations.

2. Provides a framework for integrating all your development tools into a single, consistent operating environment.

3. MS-DOS, OS/2 & Dual Mode versions are all included in the same box so *your* team can move into the 90's with the same editor.

4. Licenses are available for you, your team, division or company. The initial cost is low, and you continue to save as you add users.

MS-DOS | OS/2 | Dual Mode — All in One Package

$295 U. S. List Prices. Single User Version.

$395 Includes a Microsoft® Mouse.

Specify serial, bus or PS/2 mouse. A 5-User License (without mice) is only $1,032.50. Call for prices on larger licenses. **Visa, MC** & P.O. orders:

1-800-547-4000

30 Day Money Back Guarantee

SAGE SOFTWARE

Finally, A True Communications Tool for Applications Programmers

Dovetail™ allows your applications to transfer data between many remotes and a central PC, LAN, mini, or mainframe.

- Application level control of communications session.
- Multi-level security.
- Intrinsic file collision detection and management.
- Simultaneous multi-line capability at central PC.
- Unlimited license for remote software.
- Rock-solid error checking.

Free Remote Software!

Even our pricing is easy to work with. The remote module of Dovetail is licensed for unlimited reproduction, so upgrading the power of your application doesn't cost you a dime in royalties.

DOVETAIL COMMUNICATIONS TECHNOLOGIES, INC.

For more information about Dovetail call:
415-492-1303

CIRCLE NO. 387 ON READER SERVICE CARD

OVERLAY DRUDGERY BITES THE DUST !

Introducing Overlay Toolkit.™ So fast, so smart, so easy it practically completes overlays for you.

In 4 hours, an analyst who knew nothing of the application cut its memory requirement from 377K to a performance-tuned 198K. Here's how:

OVERLAY MASTER™ So long tedium. Hello, results! You supply your object code, it creates a tight Plink86*plus* or .RTLink response file.

OVERLAY PROBE™ Do quick "whatif" on your designs. Pinpoint programs that force up memory size.

Imagine doing overlays 90% faster. Relief for just $395. Extended memory version (for more than 600 object files),$995.

LLOYD BUSH, INC. OVERLAY TOOLKIT
156 William Street, New York, N.Y. 10038
(212) 962-4004 Fax: (212) 608-6669

Software products since 1971

1989 Lloyd Bush, Inc. Overlay Toolkit, Overlay Master and Overlay Probe are registered trademarks of Lloyd Bush, Inc.

CIRCLE NO. 344 ON READER SERVICE CARD

(continued from page 20)

These consist mainly of the "encode" and "recall" functions central to the BAM. Encode simply takes the "vecpair" corresponding to the association and adds it (with matrix add) to the current BAM. Recall "feeds" the presented pattern through the matrix (with dot products and by applying a threshold function as discussed earlier) to return another vector. We keep feeding the vectors back and forth until they stabilize to a consistent pattern association. There are also some auxiliary functions for checking the integrity of the BAM, returning its energy for a particular association (as discussed earlier), and for "uncoding" or removing an association from the BAM.

The Bam System class consists of an array of pointers to BAM matrices. Each time a BAM matrix is saturated, a new matrix is created, and the new pattern association is stored in it. The major functions are again "encode" and "recall." Encode attempts to store the pattern association in each of the BAM matrices until it succeeds. It will create a new BAM matrix if it runs out of matrices. Recall performs a BAM matrix recall operation on each of the BAM matrices. The returned association that is closest to the presented pattern and has the lowest energy relative to its matrix (as discussed earlier) is then returned as the "correct" pattern association. Another function is provided to "train" the Bam System from a specified file of pattern associations. The patterns happen to be represented as 01 strings, but this could be easily changed to whatever representation (for example, floating-point numbers, character strings) suits the specific application.

BAMs are a neural-network-based attempt at content-addressable memories

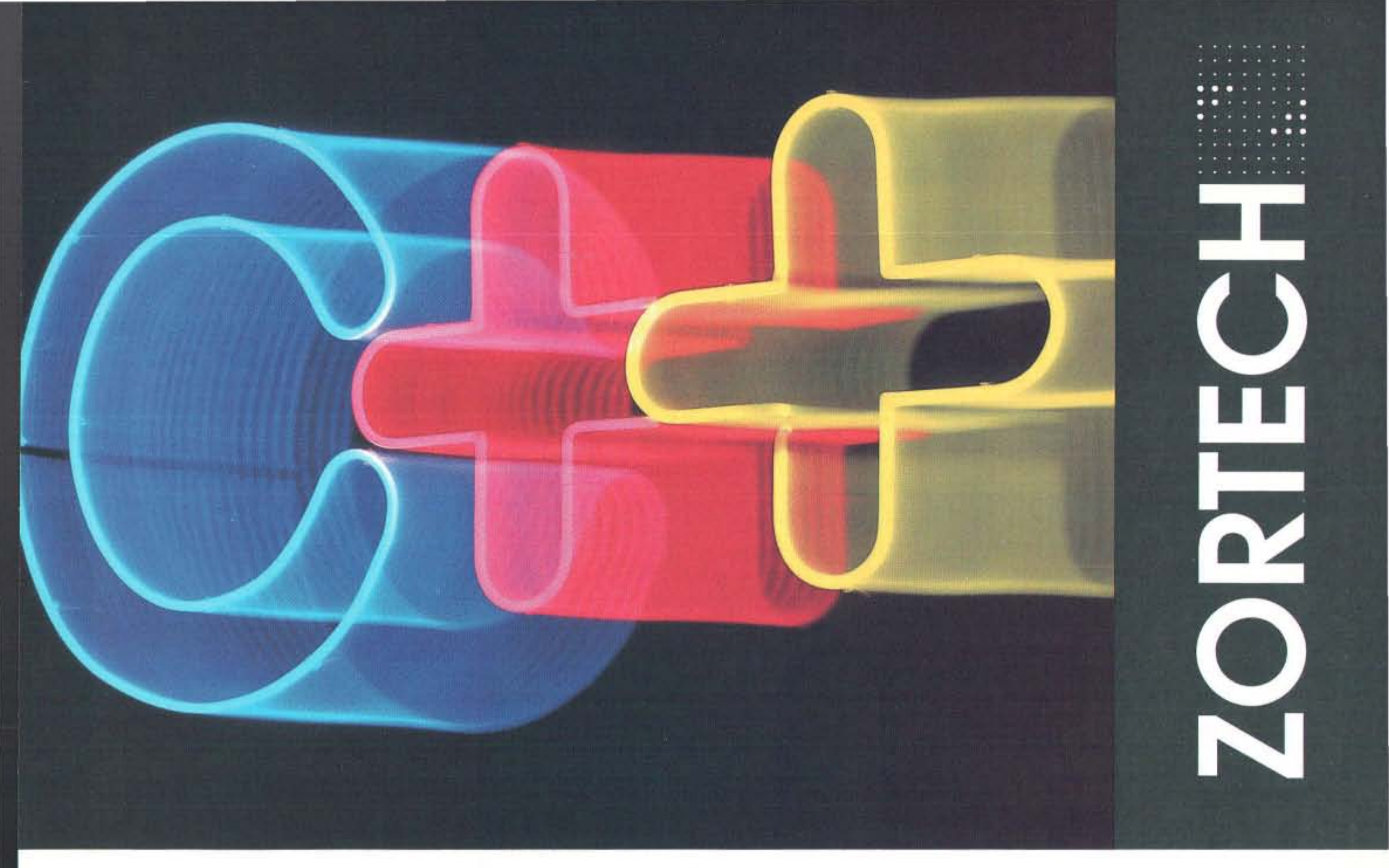
Thanks to the wonders of C++, the code is very readable. Most of the algorithms can be implemented in the same vocabulary as the theory. Take a look at it to examine the mechanics in detail. It should even be clearer (and certainly more specific) than the discussion above.

The Test Program

I've included a test program (TESTBAM.CPP, Listing Three, page 88) that demonstrates an actual running Bam System. A Bam System is created and told to "train" itself from the file TEST.FIL (see Listing Four, page 88). This file contains a set of simple "pattern pairs," represented as (0,1) strings delimited by commas — one pattern pair to a line. Once the Bam System is trained, you can enter any pattern you want (using the 01 format mentioned) and the correct pattern association will be recalled. If the pattern is slightly wrong, the correct pattern association will still most likely be recalled. The make file, TESTBAM.MK (Listing Five, page 88), shows how to construct this test program.

What Can you Do With It?

Uses of the Bam System are constrained only by your imagination. Obvious uses include optical character recognition (the pixel patterns scanned in would be associated with the actual letters), voice recognition (the acoustic pattern would be associated with the actual word), or a super spell checker (word patterns associated with phoneme-string patterns). You can use a Bam System in just about any



"Zortech C++ is one of the best MS-DOS products I've had the luck to use. I can highly recommend Zortech V2.0"

Scott Robert Ladd – Dr. Dobbs Journal – January 1990

NEW! AT&T C++ V2.0 SPECIFICATION

NEW! MS WINDOWS COMPATIBILITY

NEW! C++ SOURCE LEVEL DEBUGGER

NEW! EXPANDED C++ TOOLS

NEW! OS/2 COMPILER UPGRADE

NEW! EASIER PORTABILITY FROM MSC

Zortech is first again with the release of its new C++ V2.0 Developer's Edition featuring the AT&T C++ V2.0 specification.

New V2.0 features like Multiple Inheritance and Type Safe Linkage make this the most advanced compiler available today.

You get 1500 pages of clear, high quality, professional documentation.

Zortech C++ V2.0 makes it really easy for you to move code over from most other leading C compilers.

Zortech C++ V2.0 Developer's Edition comes with a great new environment that lets you edit, compile and debug with ease.

Zortech present another "World's First" with its new C++ Source Level Debugger for MS-DOS. Once you've used our debugger you will never want to go back to any other.

The Developer's Edition also includes a 99% ANSI compatible C compiler, seamless LIM/EMS support, C++ Graphics Shell, TSR functions, C++ Tools, Optimizer, SAA/CUA style user interface, and full standard library source code.

Please call for our color brochure.

PRICES

C++ Compiler	$199.95
C++ Debugger	$149.95
C++ Tools	$149.95
Library Source	$149.95

Save $200 – Get the Developer's Edition for only $450 (includes all the above items).

OS/2 Compiler	$CALL
C++ Video	$499.95

USA: Zortech Inc.
1165 Massachusetts Ave.
ARLINGTON MA02174
Voice: 617-646-6703
Fax: 617-643-7969

EUROPE: Zortect Ltd.
106-108 Powis Street
LONDON SE18 6LU
Voice: 44-1-316-7777
Fax: 44-1-316-4138

HOTLINE 1-800-848-8408

(continued from page 24)

application where you have a large number of "associations" that you would like to be able to recall close to instantaneously, and where some tolerance for error would be useful.

A successful application of BAM for radar signature classification was presented at the January 1990 International Joint Conference on Neural Networks (IJCNN). However, it was not a Bam System, and the implementors had to resort to various other tricks to get around capacity limitations. Several other associative memory applications appeared; but none of them were associative memory systems. They all would probably run into the capacity roadblock eventually for large data sets. Associative memories and BAMs have begun to appear implemented in VLSI, but again the capacity will prove to be a limitation for practical work. Bam Systems should have a radical effect on the usefulness of these chips.

Conclusion

Bidirectional associative memories appear to provide the content-addressable memory long sought after by computer scientists. They provide instant recall of pattern association, tolerance for error and fuzziness in the provided pattern, and global stability. However, by themselves they face some limitations. Simple BAM matrices cannot encode more pattern pairs than the smaller of their two dimensions. Some applications have inherently smaller pattern length, and, for them, matrix capacity will prove to be a severe limitation. However, the Bam System appears to overcome this problem, making associative memory a reality.

Notes

Grossberg, S. *The Adaptive Brain*, I & II. Boston, Mass.: Reidel Press, 1982.

Kohonen, T. *Self-Organization and Associative Memory*. Berlin, W. Germany: Springer-Verlag, 1977.

Kosko, B. "Bidirectional Associative Memories," IEEE Trans. Systems, Man, Cybernetics, Vol. SMC-L8, 49-60, Jan./Feb. 1988.

Rumelhart, D.E., McClelland, J.L., eds., *Parallel Distributed Processing*, I & II. Cambridge, Mass.: MIT Press, 1986.

Simpson, P.K. "Bidirectional Associative Memory Systems," *Heuristics*, 1989.

Simpson, P.K., "Associative Memory Systems," *Proceedings of the International Joint Conference on Neural Networks*, January 1990.

Availability

All source code is available on a single disk and online. To order the disk, send $14.95 (Calif. residents add sales tax) to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call 800-356-2002 (from inside Calif.) or 800-533-4372 (from outside Calif.). Please specify the issue number and format (MS-DOS, Macintosh, Kaypro). Source code is also available online through the *DDJ* Forum on CompuServe (type GO DDJ). The *DDJ* Listing Service (603-882-1599) supports 300/1200/2400 baud, 8-data bits, no parity, 1-stop bit. Press SPACEBAR when the system answers, type: listings (lowercase) at the log-in prompt.

DDJ

(Listings begin on page 84.)

Vote for your favorite feature/article.
Circle Reader Service **No. 1.**

Announcing ...

PC-lint 4.0

The Best Lint Just Got Better

PC-lint is a C source code analysis tool. It will check code more thoroughly than a C compiler and, because it looks across multiple modules, it enjoys a perspective that your compiler doesn't have.

Control your development time and cost. Find C source code errors before they find you.

Over **270 error messages**. More than **90 options** for complete customization. Suppress error messages, locally or globally, by symbol name, by message number, by filename, etc. **Easy to Use** - If you know C you already know how to use PC-lint. Check for **portability** problems. Alter size of scalars. Adjust format of error messages. Automatically generate ANSI **prototypes** for your K&R functions.

New Features in Version 4.0

- **Lint Object Modules** - incremental linting - big linting.
- **Weak Definial Checking** - macros, structs, enums, typedef's and declarations not used globally and locally. Header files not used. Externals that can be made static. Declarations that can be moved from headers. etc.
- **Library Headers** - avoid redundant prototype processing.
- **Over 70 new messages and 30 new options,** including identifier truncation clashes.
- **New "Elective Notes" message category.**
- **Unix lint compatibility**

Mainframe and Mini Programmers

Get **FlexeLint 4.0** (formerly Generic Lint) in shrouded source form for VAX/VMS, Unix, IBM VM/MVS, QNX OS-9, DG AOS/VS, etc. Requires only K&R C to compile but supports ANSI. Pricing starts at $798. Call for details.

Gimpel Software

3207 Hogarth Lane Collegeville PA 19426

(215) 584-4261 - CALL TODAY

(215)584-4266 - FAX

PC-lint is only $139 - Specify MS-DOS or OS/2

30 Day Money-back Guarantee

Quantity and educational discounts available. PA add 6% sales tax. Outside USA add $20.

PC-lint, FlexeLint and Generic Lint are trademarks of Gimpel Software.

VIDEO COURSE
Learn C++

```
class ostream {
    ...
    ostream&
      operator<<(char*);
    ostream&
      operator<<(long);
    ostream&
      operator<<(int i)
    {
      return
        *this << long(i);
    }
    ostream&
      operator<<(double);
    ostream& put(char);
    ...
```

NO RUSH!
NO TRAVEL!
NO HOTELS!
ONLY $499.95

Learn C++ Now!
The great exodus of programmers from C to C++ has begun! Since C++ builds on C, it's the easiest OOP language to learn. That's why it's called "the language of the 90's".

Why the rush?
PRODUCTIVITY! Yes, C++ programmers can write programs in less time requiring less maintenance. Large projects become much easier to manage.

Unfortunately, learning C++ can be very costly. Classroom instruction is expensive even without the travel and hotel costs. Of course, not learning will cost you even more in the long run. Now there is an alternative!

The top C++ video tutorial at the lowest possible price.
The C++ video tutorial from Zortech is the ultimate C++ training tool for work or home at only $499.95. It comes on six VHS video tapes containing 32 clear, extensive tutorials.

Used in conjunction with the concise workbook and tutorial disk, you will find everything you need for fast-track C++ tuition.

The course is generic (i.e. compiler and hardware independent) and is available with or without the award winning Zortech C++ Compiler for MS-DOS and OS/2.

Start writing C++ code within a week.
As a C programmer, you will start producing C++ code within a week of concentrated use of this course. Alternatively, spend just an hour a day watching the video and working through the suggested exercises to learn C++ in only six weeks!

The Leader in video tutorials.
If you don't already know C, you can join tens of thousands of programmers world-wide who have learnt C with the Zortech Complete C Video Course, described as:

*"An excellent bargain ...
... I heartily recommend"*
Gary Ray, PC WEEK

Save your company thousands of $$$'s.
One programmer can train for only $499.95, but you can train ten programmers for just:

C++ Course	$499.95
9 Extra Workbooks	$269.55
Total 10 students	$769.50

Yes! Only $76.95 each! (With all the FREE refresher courses you need!)

Now, Zortech with its new C++ Video Tutorial has refined the art of video tuition and presentation even further for the 1990's.

IN USA CALL:

1-800-848-8408

Only $499.95 complete
- *Six Videos with 32 lessons*
- *256 page workbook*
- *Tutorial disk*
- *Compiler & hardware independent*
- *NTSC or PAL format*
- *Tax deductible*

Don't delay, order now!
Just mail the coupon or call the order hotline for same day shipment.

USA: Zortech Inc.,
1165 Massachusetts Avenue,
ARLINGTON, MA02174
Voice: 617-646-6703
Fax: 617-643-7969

EUROPE: Zortech Ltd.,
106-108 Powis Street, LONDON
SE18 6LU
Voice: 44-1-316-7777
Fax: 44-1-316-4138

AUSTRALIA: Noble Systems
Voice: 02-564-1200
Fax: 02-564-1465

ORDER FORM

Please rush me these items:

Qty	Description	Price
___	C++ Video Tutorial	$499.95
___	Extra C++ Workbooks	$. 29.95
___	Zortech C++ Compiler	$199.95
___	Developer's Edition	$450.00

Please add $7.50 Shipping in USA
International shipping charged at cost

Name ____________
Company ____________
Address ____________
____
State ____ Zip ____________
Phone ____________
MC,VISA or CHECK ________ Expiry ______
Card No. ____________

CIRCLE NO. 258 ON READER SERVICE CARD

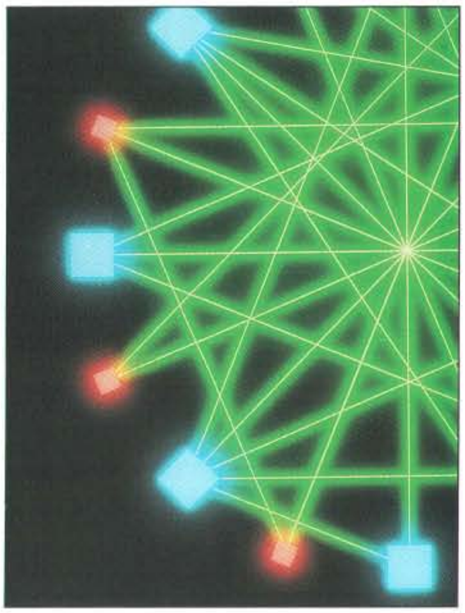
A Neural Network Instantiation Environment

Dynamically creating neural nets lets you concentrate on network response characteristics

Andrew J. Czuchry, Jr.

The automatic generation of tailored neural network architectures greatly simplifies the tedious task of putting together neural networks. Typically, an architecture is assembled by manually writing and modifying a collection of software routines; automation speeds this standard process of assembling networks. However, task simplification through the automatic generation of network architectures often implies limited flexibility when applied to real-world problems. In order to develop useful network architectures in an efficient manner, the provision of both task simplification and complete flexibility is an inherent design principle in the research environment that instantiates (dynamically creates) neural networks. Instantiation is the flexible process of automatically piecing together architectures based upon modifiable structures that represent the parameters of the assembled neural networks.

Andy earned the A.B. degree in computer science from Dartmouth College and the M.S. degree in information and computer science from the Georgia Institute of Technology. He is presently pursuing a Ph.D degree in information and computer science at the Georgia Institute of Technology. His research is supported by the Artificial Intelligence Branch of the Georgia Tech Research Institute. Andy has published several articles on topics concerning "intelligent" computer systems. He can be reached at the Georgia Institute of Technology, A. I. Branch, Georgia Tech Research Institute, 243 Baker Bldg., Atlanta, GA 30332.



The incorporation of network instantiation into an entire research environment for neural networks results in a system that provides both task simplification and complete flexibility.

Task simplification can be achieved by using a variety of knowledge-representation techniques. Complete flexibility can be maintained by strictly applying standard software-modularization techniques. The merging of these two types of techniques — knowledge representation and software modularization — provides the foundation for the instantiation process that forms the basis of a powerful neural network research environment.

In this article, I discuss the need for such an environment and describe a working version. In so doing, I describe the knowledge-representation techniques used, and the essential integration of knowledge representation and software modularization. (The model was developed on a Symbolics Lisp machine, chosen for its flexibility and power in symbolic manipulation and for its exploratory programming environment. The implementation language is Lisp.) I also present experimental results of using the environment for a test-case network, and finally, I discuss future efforts and the evolution of the environment.

System Overview

The task of generating usable neural network architectures for real-world problems is quite challenging. Basic standard networks are merely skeletons for useful systems. For example, Fukushima's neocognitron[1] is really a class of neural networks. Most often only specific instances (class elements)

Can your compiler meet the challenge?

We invite you to take an existing program and compile it using TopSpeed C. Then, compare the overall performance with the compiler you now use. If you are not 100% satisfied, return the entire package to us, and we will refund all of your money.

	TopSpeed® C Version 1.00	Microsoft® C Version 5.1	Turbo C® Version 2.0	Watcom® C Version 7.0
100% ANSI compatible*	✓			
Integrated environment	✓		✓	
Pass parameters in registers	✓			✓
Expand any function as inline code	✓			
Supports OS/2	✓	✓		
DOS Dynamic Link Libraries *(overlay code linked at load-time)*	✓			
Smart linking *(only referenced code and data linked into .EXE)*	✓			
Type-safe linking *(function parameters and memory model checked at link-time)*	✓			
Fully automatic make works across libraries	✓			
Time-sliced scheduler for multi-tasking under DOS	✓			
Short pointers in any segment	✓			
Hypertext help with library online	✓		✓	

*Written by Neil Martin of the British Standards Institution (BSI) and printed in Personal Computer World June 1989, page 241

TopSpeed®

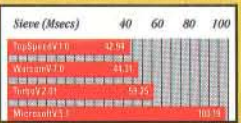
TopSpeed's seamlessly integrated multi-windowed environment.

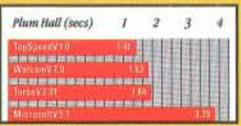
VID (Visual Interactive Debugger): a source-level, multi-windowed symbolic debugger.

In England & Europe contact:
Jensen & Partners UK Ltd.
63 Clerkenwell Road
London EC1M 5NP
Phone: (01)253-4333
Fax: (01)251-1442

C Standard Edition £149;
C Extended Edition £295;
C OS/2 Edition £370.

Call on handling & VAT charges, and TopSpeed Modula-2 product prices.





Standard Edition—
Optimizing DOS compiler, integrated source-level debugger, smart linker, automatic make, complete ANSI library, 6 memory models, BGI Interface, time-sliced scheduler, MS-DOS/BIOS interface, superset of MS and Turbo C libraries, and more.

Extended Edition—
Standard Edition plus full source code to libraries, post-mortem debugging, MS Windows support, support for DOS DLLs (overlays), assembler, disassembler, profiler, DOS call monitor, and more.

OS/2 Edition—
Extended Edition (except for DOS compiler) plus ability to generate DOS executables, full support of OS/2 Kernel & PM calls, OS/2 smart linker, fully automatic generation of DLLs, source for protected mode libraries, and more.

Benchmarks measured by Mark Hamilton, November 24, 1989. Copyright © 1989, PC Business World and CW Communications Ltd.

TopSpeed C:

Standard Edition $199
(DOS Compiler & VID)
Extended Edition $395
OS/2 Edition $495

64-page TopSpeed C Technical Specifications booklet available upon request.

Call on TopSpeed Modula-2 compiler (with objects) & toolkits.

To Order:

In the U.S., call:
1-800-543-5202
In Canada, call:
1-800-543-8452

Call on shipping & handling charges & volume discounts. VISA/MC accepted.

30-day unconditional money-back guarantee.

C

Jensen & Partners International

1101 San Antonio Road, Ste. 301
Mountain View, CA 94043
Phone: (415) 967-3200
Fax: (415) 967-3288

TopSpeed and TechKit are trademarks of Jensen & Partners International. Other brand and product names are trademarks or registered trademarks of their respective holders.

The Power of LISP Speed and Simplicity of C

You are comfortable with C, you like its simplicity and run-time performance. We have good news for you: you do not have to give up all this in order to take advantage of Lisp's tremendous power. And this is done without changing C's syntax or adding a preprocessor. For example:

```
Lisp x=NIL, y=NIL;

Set (y, MakeExp ("(C&C model)"));
Set (x, GetProp (Car(y), Cadr(y)));
```

B-Trees and **document retrieval** functions make **LispC** a good candidate for **object oriented** data base design. Functions for defining and manipulating objects (inheritance supported) are included as sample code.

In the **Regular Edition** the Lisp space is created in RAM. **$295.00**

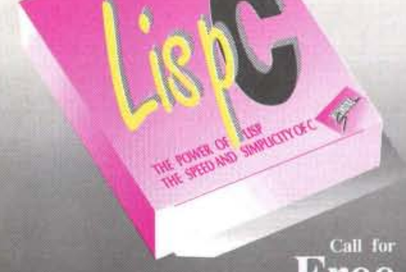
In the **Extended Edition** LispC supports over 1 Giga byte of virtual Lisp space. **$895.00**

EXTENDED EDITION 1 GIGA BYTE LISP SPACE
INCREMENTAL GARBAGE COLLECTOR

LispC's memory overhead is small, between 20 and 40 Kb. Linking LispC's object code into your application is **royalty free**.

The ***demo*** included in the package allows you to try all the LispC functions without writing one line of code. People with litle experience in Lisp will also appreciate the Introduction to Symbolic Processing. Source code available too.

An **incremental garbage collector** makes LispC ideal for **real time** applications.



Call for **Free Demo**

Available for DOS and QNX

(613) 726-6565

KLONDIKE SOFTWARE

Klondike Software Inc.
2700 Regina Street, Ottawa, Ontario
K2B 6Y1, Canada
Fax: (613) 820-8299

(continued from page 28)

are described in the literature — the skeleton for a neocognitron is a multilayer, hierarchical neural network for visual pattern recognition. It consists of a series of layers of subnetworks that are organized according to specific guidelines. The system's exact parameters (for example, the number of layers, the number of subnetworks per layer, and the size of each subnetwork) are often tailored to the problem being addressed. This tailoring is the meat on the skeleton and is determined by the application's processing requirements.

Such tailoring is evident in the differences between the architectures of the neocognitron described by Fukushima and Miyake[2] and by Fukushima.[10] Fukushima and Miyake[2] describe a seven-layer system for type-written or stylized (that is, written to meet certain specifications of consistency) numeral recognition. Each layer of the network has 24 subnetworks, except for the input layer, which is a single subnetwork layer. In contrast, the architecture of the neocognitron for hand-written numeral recognition, as described in Fukushima,[10] is a nine-layer network with 1, 12, 8, 38, 19, 35, 23, 11, and 10 subnetworks per respective layer.

In addition to these architectural differences, the setting of various internal parameters may also vary according to the application. More noise tolerance is provided by decreasing the inhibitory ("negative") weights and shrinking the number of connections per node in a subnetwork. Finer degrees of class separation are provided by increasing inhibition and increasing the number of connections between the nodes in each subnetwork. A variety of other internal parameters can be altered as well.

Given the goal of efficiently establishing useful architectures and parameter settings for real-world applications, automatic generation of neural networks based upon a flexible representation of the desired characteristics is vital. This goal has been realized through the development of the research environment described in this article. The environment dynamically creates neural networks based upon the information encoded in underlying knowledge-representation structures. The research environment automatically builds these structures based upon parametric specification of the desired characteristics of the network architecture.

For example, passing the network creation routines the network type of *neocognitron*, the layer number *9*, and the subnetwork size list of *(1, 12, 8, 38, 19, 35, 23, 11, 10)* would produce an architecture similar to the one described by Fukushima.[10] An exact match to Fukushima's architecture could be obtained through additional parameter specifications. The key point is that the research environment comprises a combination of multipurpose routines that are pieced together appropriately through the use of flexible knowledge representation structures. The environment's flexibility is maintained through the strict application of software-modularization techniques. For example, software modularization ensures that weight calculation routines can be adjusted independently of the connection calculation routines. These ideas are clarified in the following sections.

Knowledge Representation

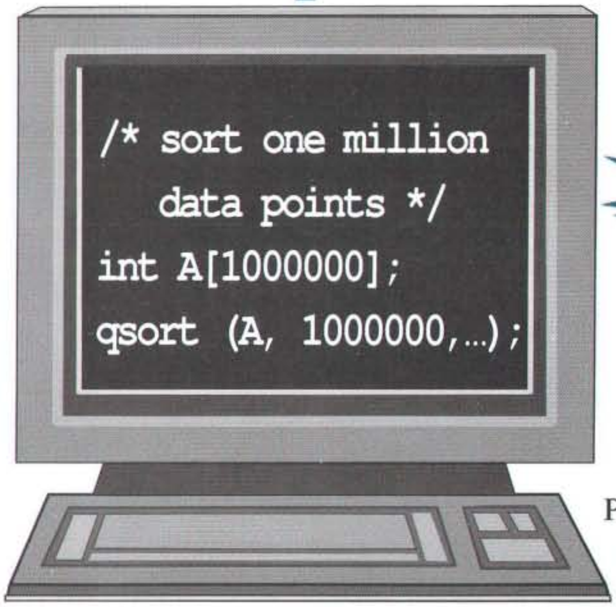
There are two fundamental ideas behind the use of knowledge representation. The first is that simple parametric changes can significantly alter the network architecture's final structure. For example, changing the size (number of nodes) in each subnetwork can greatly affect the specific connections between the nodes. This is of primary importance in networks such as the neocognitron[1] for two reasons:

1. The connections are between subnetworks rather than within subnetworks.
2. The nodes are not completely connected (that is, every node is connected to only a subset of the nodes in other subnetworks). This means that the connection architecture is heavily influenced by the size and number of subnetworks.

The second fundamental idea is that many routines for the creation of neural networks and subsequent network processing are common to entirely different architectures. As a result, these routines can be reused and, to some degree, tailored automatically by combining and adapting the modules. The realization of a research environment that automatically generates flexible neural network architectures has, thus, been based upon a knowledge representation in which every structure "carries around with it" all the local information for piecing itself into the network puzzle and for subsequently computing/processing data once the architecture is assembled.

Three main knowledge structures — *NET*s, *LAYER*s, and *PLANE*s — collectively compose the knowledge representation. These structures are presented in Listing One, page 93. The *NET* structure consists of a list of one or more layers and a variety of local parameters specific to the global processing of the particular type of architecture (for example, the vigilance parameter in ART.[3,5]) *LAYER* encodes a list of subnet-

MS-DOS 386•486
Can your C compiler handle this?



```
/* sort one million
   data points */
int A[1000000];
qsort (A, 1000000,...);
```

MetaWare's High C® and Professional Pascal™ — the *first* C and Pascal compilers to generate 386 protected-mode code for MS-DOS. And the compilers themselves can run in protected mode, giving you the power to create large programs.

Does your current compiler give you 32-bit code? Arrays up to 4 gigabytes? 387 in-line transcendentals and Weitek support? MetaWare's ANSI compilers and Phar Lap's 386|DOS-Extender let you expand to the 32-bit registers and address space of the Intel 386 and 486.

New Release! Version 1.6 includes

- ***Expanded MetaWare library with many Microsoft-compatible functions***
- ***New make utility***
- ***One-step compile and link driver***
- ***Sterling Castle's BlackStar 386 library***
- ***Sterling Castle's BPT Plus b-tree***
- ***GFX/386 graphics routines***
- ***EC and MicroEMACS editors***
- ***DOS Helper UNIX-like utilities***

Practically every 386 protected-mode MS-DOS program available is done with MetaWare's High C or Professional Pascal.

Industry leaders such as

- ***Autodesk – AutoCAD***
- ***Borland – Paradox 386***
- ***Fox Software – FoxBase+/386***
- ***IBM – Interleaf Publisher***
- ***Wolfram Research – Mathematica***

and more use MetaWare's compilers to get dramatic increases in performance. Don't let *your* 386 power go to waste!

Call now for your 386 compilers and DOS extenders: (408) 429-6382.

MetaWare® INCORPORATED

Compiler Products for Professional Software Developers

2161 Delaware Avenue • Santa Cruz, CA 95060-5706 • Phone (408) 429-META(6382) • FAX (408) 429-WARE(9273)

** MIPS magazine, October, 1989. See articles on 80486, YARC 29000, and 386 Great Performers.*

CIRCLE NO. 158 ON READER SERVICE CARD

(continued from page 30)

works and the connections between layers (e.g., a list of the connections from each subnetwork in the present layer to the subnetworks of the preceding layer). Local parameters, such as inhibition constants or gain constants, are also stored within the layer. The type of the layer (for example, S or C for the neocognitron[1] and F_1 or F_2 in ART[3,5]) is also recorded. A pointer to the previous layer is provided so that the routines can "get around" in the network. *PLANE*, a subnetwork, is used to store the connections within the subnetwork. The weights for both inter- and intraplane connections are also recorded in the *PLANE* structure. A *size* parameter for the plane is used for instantiation and is locally encoded. A pointer to the layer of which this plane is a part is stored as well. The actual nodes (cells) or processing elements are stored as an array, which is used to record output activation values.

In the future, this array will be extended so that each node is itself a knowledge structure. In this way, the activation functions, output functions, and local node parameters can be maintained locally. Such an extended representation will further increase the environment's flexibility by providing for the adjustment of input and output activation functions within the overall architecture and, thereby, will extend the standard of common activation functions for each cell in the subnetwork. This standard for specific activation functions for the entire subnetwork is not a severe restriction. However, the extended representation will support novel research endeavors and, thus, could prove to be extremely valuable.

The information contained in these knowledge structures is stored at the time of network instantiation and is utilized as the computational map by the processing routines. The structures thus dynamically control the routines to be called, the data to be passed, and the amount of processing to be performed. Each structure has been designed to carry locally all the information necessary to direct processing through the contents of the structure itself rather than through a priori routines. This advantage increases processing flexibility. In adapting the flow of processing, no routines need to be altered; only the information in the structures is modified.

Modularity

Great care has been taken to ensure software modularity. The significance of this is apparent upon analysis of the power obtained by the integration of knowledge representation and software modularization. Before discussing the integration, however, I will briefly describe the modularity.

Software modularity has been preserved in all three of the main phases of neural network applications: instantiation, processing, and training. The network instantiation routines, highlighted in Listing Two, page 93, are the pieces that are meshed to dynamically create network architectures. Instantiation is obviously the first step in the use of neural networks for any application. The instantiation process proceeds from generic net-level creation routines to specific layer-level creation routines that are tailored to the specific type of network. Finally, it proceeds to generic plane-level routines that perform connection and connection weight calculations in addition to creation of the plane itself. Upon completion of the instantiation process, the network is ready for training.

Listing Three, page 96, indicates the training routines. These routines are used to perform the processing all the way from the network level down to the level of the individual cell. Listing Three is abbreviated, however, and depicts the routines only down to the initial processing at the plane level. Subsequent processing occurs at the plane, connection, and cell levels. After training, the network can be used for identification tasks. Identification functions at the network level are presented in Listing Four, page 98. Further processing occurs at the layer and plane levels but is not included here.

Integration

The knowledge representation structures function as generic placeholders in which data about instantiated neural networks is recorded. As mentioned previously, the instantiation process dynamically produces an entire network based on the parametric specification of the desired characteristics. Instantiation begins by calling *CREATE-NET* (see Listing Two) and passing it the appropriate parameters for the desired network.

Two examples of the parametric settings and function calls for different versions of a neocognitron are depicted in Listing Five, page 98. Evaluating **neocognitron-net**, (that is, *(eval *neocognitron-net*)*) returns a *NET* structure (see Listing One) that contains an instantiated network meeting the characteristics specified in the parameters recorded in the **neocognitron-net** variable. More specifically, a seven-layer network would be created. The input layer would contain one subnetwork (plane), and each of the other layers would contain 24 planes. The plane in the input layer would contain a 16×16 array of nodes (cells). Each of the planes in the next layer would contain a 16×16 array of cells. Subsequent layers would be composed of planes with 10×10, 8×8, 6×6, 2×2, and 1×1 arrays of cells, respectively.

Each cell in a plane would have a "square" projection pattern; cells are connected to other cells that occupy a corresponding square area in another plane. Connections from the first layer to the input layer would cover a 5×5 array. Connections from the second layer to the first would also cover a 5×5 array, and similarly for all the connections up to and including the connections from the sixth layer to the fifth layer. Connections between the last (seventh) and the sixth layer, however, would cover a 2×2 array. Each cell, x_i, would thus become an input to multiple other cells, x_j, and each x_j would receive inputs from many different x_k cells.

Each of these connections has associated with it a connection weight. Within the knowledge representation, the connection weights are stored separately from the connections themselves so as to provide for adaptation of the weights independently of the connection structure. In addition to these elements common to all neural networks (that is, one or more layers, one or more subnetworks per layer, individual cells, connections, and connection weights), a variety of other parameters significant for the neocognitron would be set as indicated in the **neocognitron-net** variable in Listing Five.

An additional comment about the assignment of connections is important. The exact connection patterns are calculated based upon the size of the sending and receiving planes and the size of the projection area. The instantiation routines are structured such that the appropriate connections are computed based upon the parameters passed to the respective routines. For example, if each element in a 5×5 network is to project into a 3×4 network such that each element has a 2×2 projection and all of the 3×4 network elements are covered, then the system would compute that the element in position (0,0) of the 5×5 network would be connected to the elements in positions (0,0), (0,1), (1,0), and (1,1) of the 3×4 network (see Figure 1). The element in position (4,4) of the 5×5 network would be connected to the elements in positions (1,2), (1,3), (2,2), and (2,3) of the 3×4 network. A variety of standard connection architectures have been presented in

POSE™... *Your First CASE TOOL With A Risk Free, Money Back Guarantee.*

No Other CASE Vendor Makes This Offer.

Introducing POSE™... a modular set of integrated, PC based, systems analysis and design tools that we guarantee you'll like.

Innovative – The first of its kind to provide instant analysis at critical intersections of the data and process models.

Comprehensive – POSE supports an extensive range of analysis and design techniques such as Data Modelling, Dataflow and Decomposition Diagramming, Program Structure Charting and Logical Access Path Analysis.

Powerful – Advanced graphics capabilities easily manage large diagrams at sub-second speeds.

Easy to use – Consistent user interface across all modules utilizes pull-down/pop-up menus, on-screen icons and color graphics.

Affordable – POSE consists of nine separately priced modules which can be integrated as you grow. Buy only *what* you need, *when* you need it.

POSE... for a Picture of Productivity Like You've Never Seen Before– Guaranteed!

"If you have been listening to sales pitches for $10,000.00 CASE products, you should check this one out."

–Ed Yourdon
American Programmer
9-88

POSE modules may be purchased individually for $595, or purchased as tool kits...

Data Model Tool Kit	Process Model Tool Kit
Data Base Aid	Decomposition Diagrammer
Logical Database Designer	Data Flow Diagrammer
Data Model Normalizer	Action Chart Diagrammer
Data Model Diagrammer	Structure Chart Diagrammer
$1195 *Includes the above 4 modules*	**$1195** *Includes the above 4 modules*

POSE **Computer Systems Advisers, Inc.**
50 Tice Boulevard
Woodcliff Lake
New Jersey 07675

...for Picture Oriented Software Engineering.

CALL 1-800-537-4262

CIRCLE NO. 413 ON READER SERVICE CARD

Introductory pricing subject to change without notice.

AD J040

(continued from page 32)

the neural network literature — for example ART,[3,5,6,7] back propagation,[8] Hopfield networks,[9] and the neocognitron.[1] Because the connection calculation routines are parameterized and actually calculate the connection patterns, arbitrary algorithmically expressed connection patterns can be realized.

Experimental Results

To investigate the viability of the research environment presented in this article, a standard neural network architecture was chosen to test the environment's instantiation, training, and identification capabilities. The network chosen was the neocognitron[1] because of its large size and the complexity of the connection architecture. More specifically, the architecture presented by Fukushima and Miyake[2] was reproduced and is characterized by **neocognitron-net** in Listing Five. For this version of a neocognitron, there are a total of more than 2.3M connections, each with its own weighting factor, between the 10K cells and 145 planes in the network. Additionally, several parameters interact and affect the behavior of the network. For example, each of the layers (excluding the input layer) has an intensity-of-inhibition parameter to control the amount of noise tolerated in matching a pattern; this parameter interacts with both the excitatory and inhibitory weights as an output is computed for a particular network cell.

Instantiation of the network, described by the **neocognitron-net** variable in Listing Five, and subsequent training and identification testing yielded significant results:

1. Different patterns produce different excitation patterns within the network (see Figures 2 and 3).
2. Training the network alters its excitation patterns (see Figures 3 and 4).
3. After training, only a single cell fires at the recognition layer in response to different stimulus patterns (Figure 4).
4. Appropriate clusterings are achieved for multiple versions of various numeric characters (see Figures 4, 5, and 6).

The input pattern is depicted at the base of each figure in Figures 2 through 5. Each layer is represented by the double row of squares (planes), which are respectively labeled S_1, C_1, S_2, C_2, S_3, and C_3 on the right-hand edge of the figures. The colored areas within each square represent the output activities of the corresponding nodes (cells). The color scale is shown on the left-hand edge of the figures and indicates that

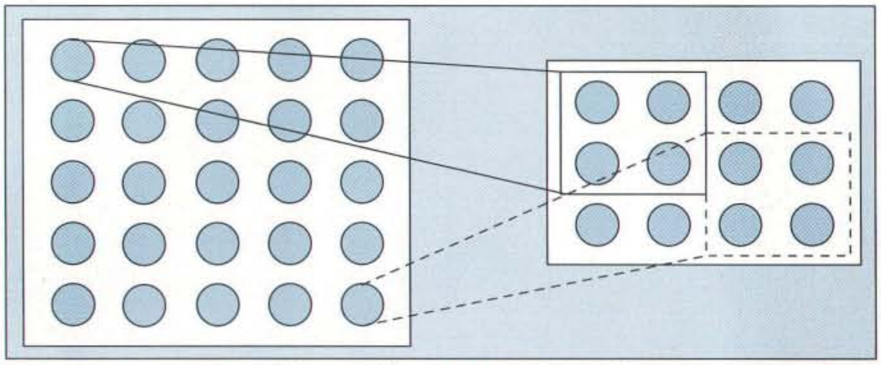


Figure 1: *Connections calculated for 2× 2 projections from a 5× 5 network into a 3× 4 network*

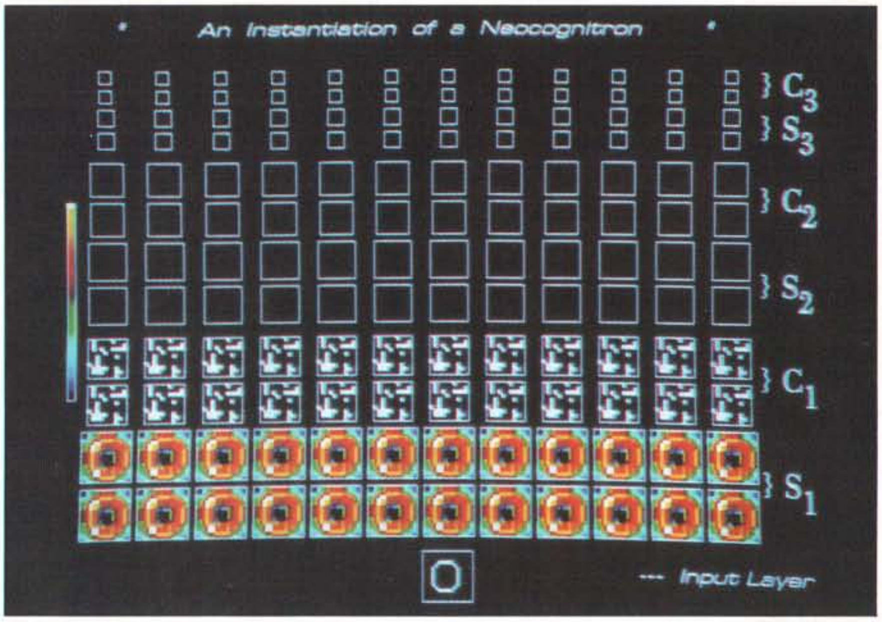


Figure 2: *Response characteristics of an instantiated neocognitron to the input pattern 0 before training*

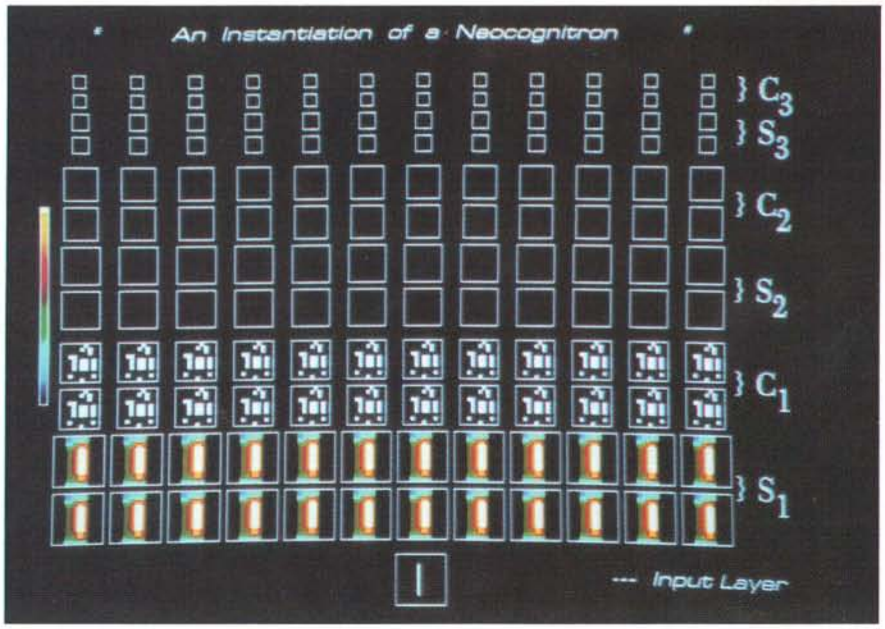


Figure 3: *Response characteristics of an instantiated neocognitron to the input pattern 1 before training*

activity ranges from a low level of black to blue, to green, to red, to yellow, to a high level of white. These pictures are produced within the research environment as a useful utility for qualitatively observing the results of the particular instantiated architecture. As depicted herein, the results correlated well with those presented by Fukushima and Miyake.[2]

Future Efforts

The research environment and corresponding instantiation of neural networks have many possible applications. From a general perspective, such applications encompass both new-model development and the analysis of standard network models. More specifically, one of the motivating ideas behind this research has been that of using digitized images as training patterns. The hypothesis is that network models such as the neocognitron should theoretically be able to extract "useful" information from such images. For example, through training an instantiated network using a variety of images that contain a tree, the network should extract the common pattern of the tree and, thus, be able to indicate the presence of a tree in subsequent test images.

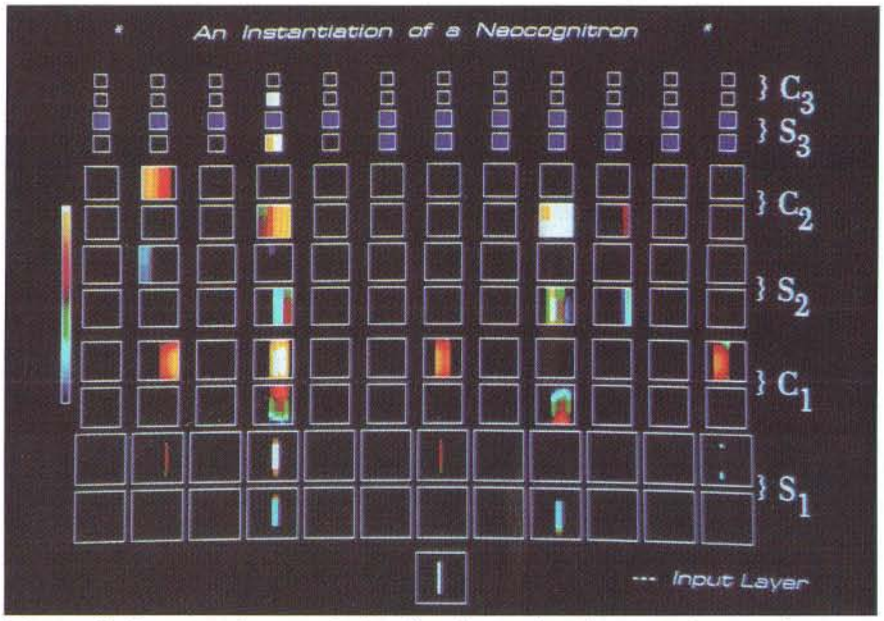


Figure 4: *Response characteristics of an instantiated neocognitron to the input pattern 1 after training*

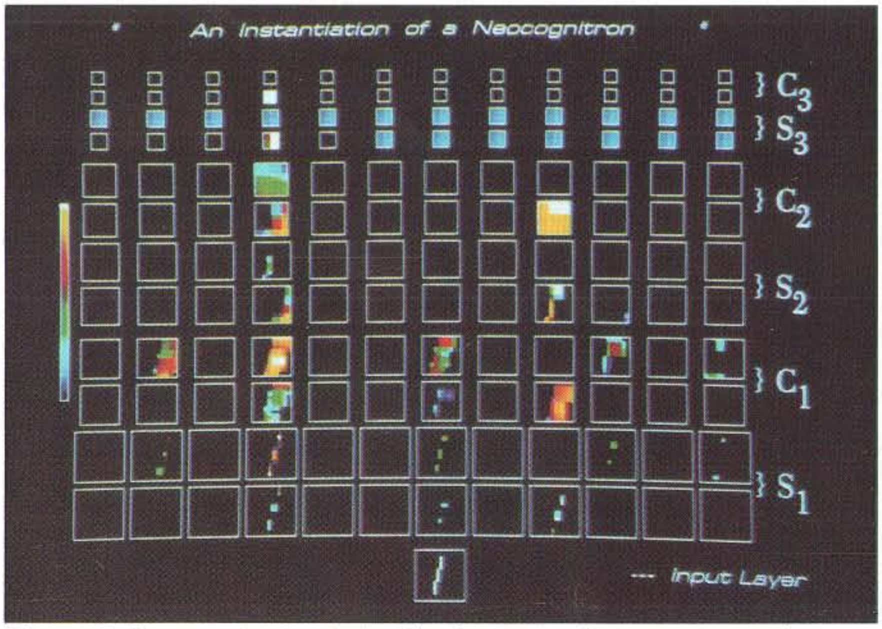


Figure 5: *Response characteristics of an instantiated neocognitron to the input pattern of a slanted 1 after training*

NEW PRODUCT!

Translate AWK to C

Now you can have the execution speed and security you always wanted with your **awk** scripts. Code faster in **awk** for simplicity. Then use the ***AWKCC*** translator to compile and execute for speed and security. ***AWKCC*** is transparent too: **awk** scripts that execute in an interpretative mode will produce the same results in compiled mode - only *much* faster!

the Korn Shell

The *best* shell for UNIX Systems. Now shipping the ***ksh88*** version with **ASPEN's** *kforms* extension. The ASPEN KORN Shell delivers more power per keystroke - save time, save money, boost productivity, and remain compatible with Bourne shell scripts. Display forms and accept data input with *kforms*, the ***ksh88*** forms capability extension.

BOTH AVAILABLE FOR:
386 UNIX & Xenix, Sun 386i; 3B1/7300 Sun3, Sun4; 3B2; IBM RT; NCR Tower; Sequent/386; Plexus/386

BOOKS:

The KORNSHELL Book
by *Bolsky & Korn* - $36.00

The AWK Programming Language
by *Aho, Kernighan, Weinberger* - $29.00

ASPEN Technologies, Inc.
PO Box 5727
Parsippany, NJ 07054

Telephone: **201-316-0866**

FAX: 201-316-5781

Check or Money Order, **VISA** and **MC** accepted. Please include $5.00 for Shipping & Handling to the U.S. and Canada. $30.00 Overseas; (N.J. residents add 6% Sales Tax). Money Back Guarantee.

kforms is a trademark of ASPEN Technologies, Inc. XENIX is a registered trademark of Microsoft Corp. UNIX is a registered trademark of AT&T.

CIRCLE NO. 325 ON READER SERVICE CARD

A possible future research effort would investigate the size of various networks required to actually perform such recognition and to characterize any additional requirements (for example, use of Grossberg's Boundary Contour/Feature Contour System [S. Grossberg and E. Mingolla, "Neural dynamics of perceptual grouping: Textures, boundaries, and emergent segmentations," Perception & Psychophysics, 38 (1985), 141 – 171.] as a preprocessor to simplify processing within an appropriate version of a neocognitron). Additionally, recent extensions to the neocognitron's architecture (that is, feedback between layers[3]) could be incorporated into the currently instantiated neocognitrons and could possibly provide for segmentation of trees within the test images after training has occurred. A significant amount of work would be required to obtain such results, but a real possibility of attaining them does exist.

On the front of run-time analysis of instantiated networks, the speed and versatility of processing related to the implementation of the environment should be considered. As mentioned, the present implementation, which was developed for in-house use, was developed on a Symbolics Lisp machine. The lisp machine was chosen for its flexibility and power in symbolic manipulation and for its exploratory programming environment. The implementation language is Lisp. In order to enhance processing speed, there is a plan to port the environment to a Sun 4/280. Although the environment is currently organized to dynamically create Fukushima's neocognitron,[2] its versatility will be tested by instantiating additional neural network models.

Conclusion

The main virtue of the environment described here is that it frees the user/programmer/researcher from the need to write programs that assemble neural networks; the environment automatically generates flexible neural network architectures based upon parametric specifications. Flexibility is achieved through the integration of knowledge-representation and standard software-modularization techniques. Together, these two types of techniques form the powerful basis of a research environment for neural networks.

The mechanisms through which the power is harnessed and utilized are the heart of this article. The key idea is that a knowledge representation has been developed so that each element of a neural network carries around with it all the information necessary for local processing (for example, what processing to perform and where to get the inputs). Because this information storage is consistent from the node level to the network level, the entire network is executed without the need for global routines to encode its structure. Generic routines become specialized processors as they are adapted by the contents of the knowledge structures.

The viability of such an environment has been demonstrated through the instantiation and testing of a standard network architecture, the neocognitron. The results correlate accurately with those of an analogous network described by Fukushima and Miyake. The present work suggests many significant applications, some of which are currently under investigation.

Knowledge representation and software modularization are key tools ideally suited for the empirical analysis of neural networks. Wrapping an environment around these fundamental tools facilitates concentration on network-response characteristics rather than on monotonous debugging of specialized routines that encode network architectures.

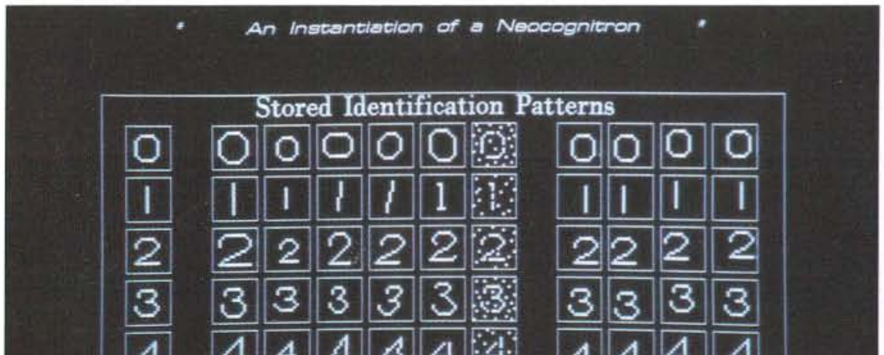

An Instantiation of a Neocognitron
Stored Identification Patterns
Clean
Distorted
Shifted

***Figure 6:** Patterns that are properly recognized by an instantiated neocognitron. The neocognitron was first trained on the patterns in the leftmost column*

Acknowledgments

This work has been performed under the guidance and with the support of John Gilmore, head of the Artificial Intelligence Branch at the Georgia Tech Research Institute. Additional contributions to the initial design and development of this environment have been made by Harold Forbes and Steven Strader. I am indebted to Diane Czuchry for her assistance in the preparation of this manuscript.

Notes

1. K. Fukushima, "Neocognitron: A self-organizing neural network for a mechanism of pattern recognition unaffecte by shift in position," *Biological Cyb netics* 36 (1980) : 193 – 202.
2. K. Fukushima, and S. Miyake, " cognitron: A new algorithm for recognition tolerant of deformat shifts in position," *Pattern Re* 15 (1982) : 455 – 469.
3. G. Carpenter, and S. Grossberg, "A massively parallel architecture for a self-organizing neural pattern recognition machine," *Computer Vision Graphics Image Processing* 37(1) (1987) : 54 – 115.
4. K. Fukushima, "A neural network for visual pattern recognition," *IEEE Computer* 21(3) (March 1988) : 65 – 75.
5. G. Carpenter, and S. Grossberg, "ART2: Self-organization of stable category recognition codes for analog input patterns," *Applied Optics*, 26 (1987) : 4919 – 4930.
6. S. Grossberg, "Adaptive pattern classification and universal recoding: I. Parallel development and coding of neural feature detectors," *Biological Cybernetics* 23 (1976) : 121 – 134.
7. S. Grossberg, "Adaptive pattern classification and universal recoding: II. Feedback, expectation, olfaction, illusions, *Biological Cybernetics* 23 (1976) : 187 – 202.
8. D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing* 318-362. (Cambridge, Mass.: MIT Press, 1986.)
9. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Poc. Nat. Academy Sci, USA* 79 (1982) : 2554 – 2558.
10. K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural Networks* 1 : (1988) 119 – 130.

DDJ

(Listings begin on page 93.)

Vote for your favorite feature/article.
Circle Reader Service **No. 2.**

Everything You Ever Wanted In UNIX. And Less. $99.95.*

COHERENT
A Multi-User, Multi-Tasking Operating System for the IBM PC/AT and Compatible 286 or 386 Based Computers.

OK. We know it's hard to believe. So just consider this. Coherent™ is a virtual clone of UNIX. But it was developed independently by Mark Williams Company. Which means we don't pay hundreds of dollars per copy in licensing fees.

What's more, Coherent embodies the original tenet of UNIX: small is beautiful. This simple fact leads to a whole host of both cost and performance advantages for Coherent. So read on, because there's a lot more to Coherent than its price.

SMALLER, FASTER...BETTER.

Everybody appreciates a good deal. But what is it that makes small so great?

For one thing, Coherent gives you UNIX capabilities on a machine you can actually afford. Requiring only 10 megabytes of disk space, Coherent can reside with DOS. So you can keep all your DOS applications and move up to Coherent. You can also have it running faster, learn it faster and get faster overall performance. All because Coherent is small. Sounds beautiful, doesn't it?

But small wouldn't be so great if it didn't do the job it was meant to do.

EVERYTHING UNIX WAS MEANT TO DO.

Like the original UNIX, Coherent is a powerful multi-user, multi-tasking development system. With a complete UNIX-compatible kernel which makes a vast world of UNIX software available including over a gigabyte of public domain software.

Coherent also comes with Lex and Yacc, a complete C compiler and a full set of nearly 200 UNIX commands including text processing, program development, administrative and maintenance commands.

And with UUCP, the UNIX to UNIX Communication Program that connects you to a world-wide network of free software, news and millions of users. All for the cost of a phone call.

We could go on, but stop we must to get in a few more very important points.

EXPERIENCE, SUPPORT AND GUARANTEES.

Wondering how something as good as Coherent could come from nowhere? Well it didn't. It came from Mark Williams Company, people who've developed C compilers for DEC, Intel, Wang and thousands of professional programmers.

We make all this experience available to users through complete technical support via telephone. And from the original system developers, too!

Yes, we know $99.95 may still be hard to believe. But we've made it fool-proof to find out for yourself. With a 60-day money-back no-hassles guarantee.

You have to be more than just a little curious about Coherent by now. So why not just do it? Pick up that phone and order today.

You'll be on your way to having everything you ever wanted in UNIX. And for a lot less than you ever expected.

LESS IS MORE!	Coherent For the IBM-PC/AT and compatible 286 or 386 based machines.	Santa Cruz Operations XENIX 286, Version 2.3.2
No. of Manuals	1	8
No. of Disks	5	21
Kernel Size	64K	198K
Install Time	20-30 min.	3-4 hours
Suggested Disk Space	10 meg	30 meg
Recommended Memory	640K	1-2 meg
Performance*	38.7 sec	100.3 sec
Price	$99.95	$1495.00

*Byte Execl benchmark, 1000 iterations on 20 MHZ 386.

1-800-MARK WMS
(1-800-627-5967 or 1-708-689-2300)
60-DAY MONEY BACK GUARANTEE!

Mark Williams Company
601 N. Skokie Highway
Lake Bluff, IL 60044

*Special introductory price good through July 31, 1990. Coherent is a trademark of Mark Williams Company. UNIX is a trademark of AT&T. XENIX is a trademark of Microsoft.

Untangling Neural Nets

When is one model better than another?

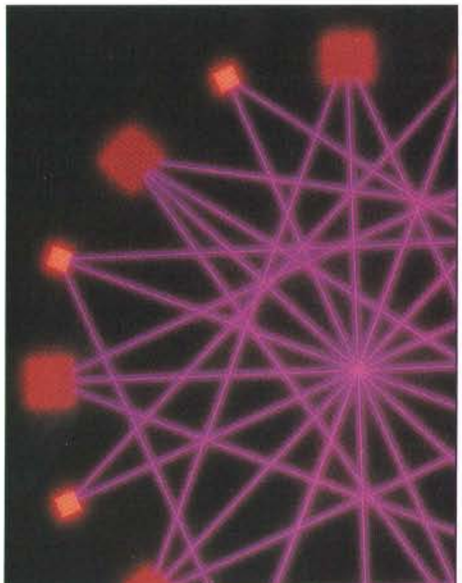
Jeannette "Jet" Lawrence

Neural networks, which are formed by simulated neurons connected together much the same way the brain's neurons are, are able to associate and generalize without rules. They have been used to classify undersea sonar returns, speech, and handwriting, predict financial trends, evaluate personnel data, control robot arms, model cognitive phenomena, and much more.

The kinds of problems best solved by neural networks are also those that people do well: Association, evaluation, and pattern recognition. Neural networks also handle problems that are difficult to compute and do not require perfect answers — just quick, good answers. This is especially true in real-time robotics or industrial controller applications.

Other appropriate applications are predicting behavior and analyzing large amounts of data, such as in stock market forecasting and consumer loan analysis. New applications under development include simple vision systems, weather forecasting, assistance in medical diagnosis, and estimation of the worth of insurance claims.

Jeannette (Jet) Lawrence is technical publications manager at California Scientific Software, and the author of their 1989 publication Introduction to Neural Networks. *She can be contacted at 160 E. Montecito #E, Sierra Madre, CA 91024.*



A neural network is not always the best solution for certain problems. They are poor at precise calculations and serial processing, nor are they able to predict or recognize anything that does not inherently contain some sort of pattern. This is why, for example, a neural net cannot predict the lottery, because a lottery is by definition a random process.

It is unlikely that a neural network could be built that has the capacity to think as well as a person does for two reasons: Neural networks are terrible at deduction (logical thinking), and the human brain is too massively complex to simulate completely. A human brain contains about 100 billion neurons, each of which connects to about 10,000 other neurons.

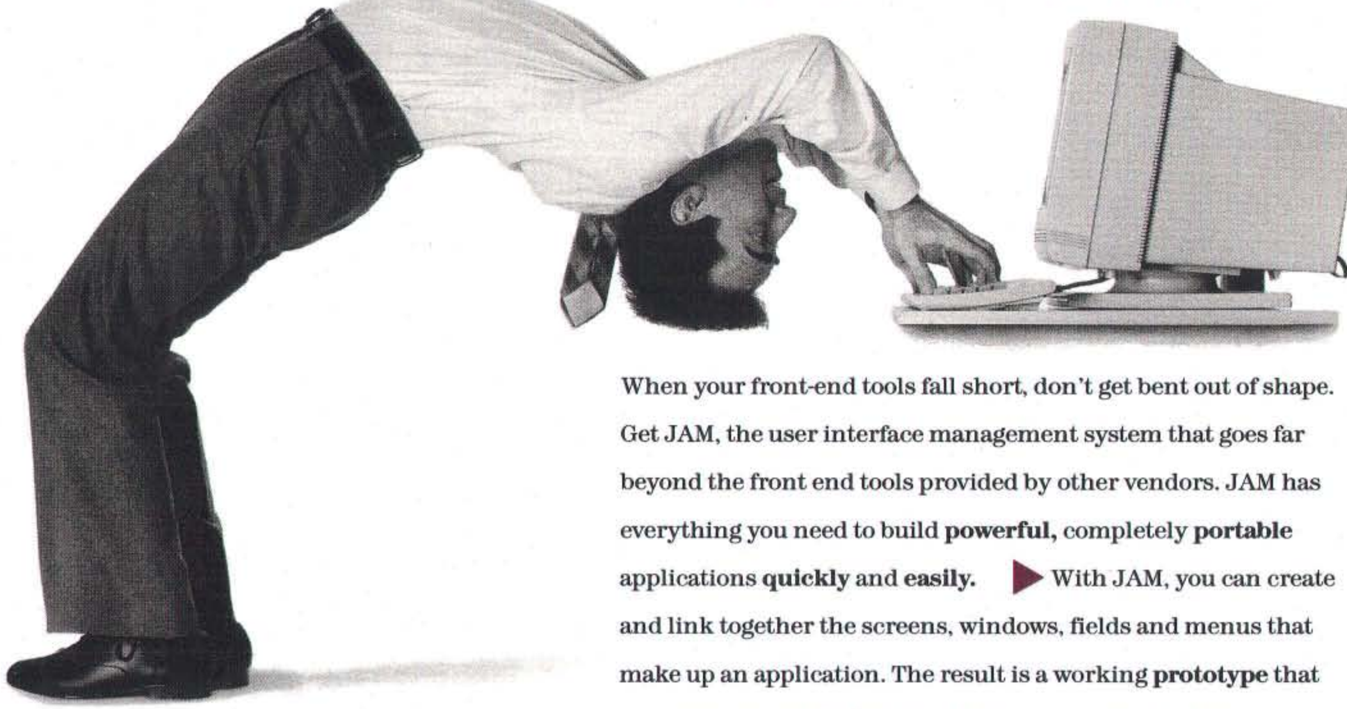
A brief look at the general structure and operation of neural networks will help explain the limits of neural networks abilities. There are many types of neural networks, but all have three things in common: Distributed processing elements (neurons), the connections between them (network topology), and the learning rule. These three aspects together constitute the neural-network paradigm.

The Formal Model of a Neuron

Artificial neurons are also known as processing elements, neurodes, units, or cells. Figure 1 shows the canonical model of a neuron. Each neuron receives the output signals from many other neurons. The point where two neurons communicate is called a "connection." This neural connection is analogous to a biological synapse in the mammalian brain. A neuron calculates its output by finding the weighted sum of its inputs. The strength of a particular connection, called its weight, is noted w_{ij}, where i is the receiving neuron and j is the sending neuron.

At any point in time (t), the activation function, adds up the weighted

BENDING OVER BACKWARDS WON'T GET YOUR APPLICATIONS UP AND RUNNING



When your front-end tools fall short, don't get bent out of shape. Get JAM, the user interface management system that goes far beyond the front end tools provided by other vendors. JAM has everything you need to build **powerful,** completely **portable** applications **quickly** and **easily.** ▶ With JAM, you can create and link together the screens, windows, fields and menus that make up an application. The result is a working **prototype** that incorporates special features like extensive field edits and validations, context-sensitive help, pop-up windows and pull-down menus—all **without coding.** ▶ And if you're working with databases, JAM's database interface, JAM/DB*i*, links your applications to many popular databases. Access is implemented directly in JYACC's procedural language using SQL. ▶ JAM applications are **fast** because processing routines are written in standard third generation languages. And, you'll never have to worry about converting your applications to a new system, because JAM is hardware, operating system and database **independent.** ▶ So, if your development tools have you turned upside down, it's time you turned to the experts. JAM, by JYACC. Specialists in application development. ▶ To receive a *FREE* demo diskette and find out more about the JAM family of software products, call our toll-free hotline today!

- ✓ Supports MS/DOS,® OS/2,® XENIX,® UNIX,® VMS,™ Primos,® AOS/VS,® VOS,® MPE/XL,® iRMX®
- ✓ Links to Informix,® INGRES,® Oracle,® Rdb,® Share Base,™ SQLBase,® SQL Server,® Sybase,® xdb®

800-458-3313

(In NY, call 212/267-7722 or FAX 212/608-6753)

JYACC

JAM

Authorized Distributors: Australia: 03 899 0063 Belgium: 3 03 829 05 81 England: 04 91 576 466 Finland: 0 452 3400 Israel: 02 811462 Italy: 02 25 52 65 2 Sweden: 8 96 10 10 Switzerland: 021 825 35 85 West Germany: 040 79 70 07 18

CIRCLE NO. 275 ON READER SERVICE CARD

(continued from page 38)

inputs to produce an activation value $a_i(t)$. In most models, input signals can either be excitatory or inhibitory, that is, they either tend to make the neuron fire or tend to suppress its firing. This value is passed through an output (or transfer) function f_i, which produces the actual output for that neuron for that time, $o_i(t)$.

After summation, the net input of the neuron is combined with the previous state of the neuron to produce a new activation value. In the simplest models, the activation function is the weighted sum of the neuron's inputs; the previous state is not taken into account. In more complicated models, the activation function also uses the previous output of the neuron, so that the neuron can self-excite. These activation functions slowly decay over time; an excited state slowly returns to an inactive level. Sometimes the activation function is stochastic, that is, it includes a random noise factor.

The transfer function of a neuron defines how the activation value is output. The earliest models used a linear transfer function. However, certain problems are not entirely reducible by purely linear methods. The threshold transfer function is the simplest of the nonlinear models. This function is an all-or-nothing function; if the input is greater than some fixed amount (the threshold), the neuron will output a 1; if the value is below the threshold, the neuron will output a 0.

Sometimes the transfer function is a saturation type of function: More excitation above some maximum firing level has no further effect. A particularly useful transfer function is called the "sigmoid function," which has a high- and a low-saturation limit and a proportionality range in between. This function is 0 when the activation value is a large negative number. The sigmoid function is 1 when the activation value is a large positive number and makes a smooth transition in between.

The behavior of the network depends heavily on the way the neurons are connected. In most models, the individual neurons are grouped into layers so that the output from each neuron

The transfer function of a neuron defines how the activation value is output

in one layer is fully interconnected with the inputs of all the neurons in the next layer. A network may include inhibitory connections from one neuron to the rest of the neurons in the same layer called "lateral inhibition." Sometimes a network has such strong lateral inhibition that only one neuron in a layer, usually the output layer, can be activated at a time. This effect of minimizing the number of active neurons is known as "competition." In a feed-forward network, neurons in a given layer do not take inputs from subsequent layers or from layers prior to the

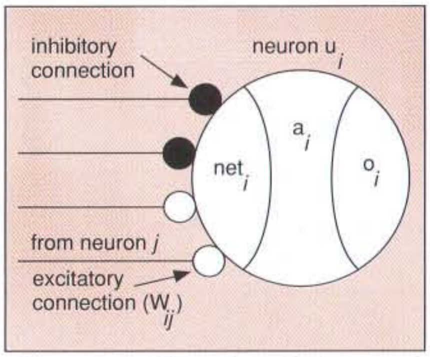


***Figure 1:** The formal model of a neural-network processing element*

immediately previous layer. Also, the neurons in a feed-forward network usually do not connect to each other. The back propagation network typically has three feed-forward layers: Input, hidden, and output. Feedback models additionally include connections from the outputs of one layer to the inputs of the same or a previous layer.

A neural network learns by adapting to changes in the input. This is accomplished through changes in the weights as the network gains experience. The learning rule is the very heart of a neural network; it determines how the weights are adjusted as the neural network gains experience. Of the numerous learning rules in use, the most well-known are Hebb's Rule and the Delta Rule. Nearly all other rules are variations of these two.

More than 30 years ago, Donald O. Hebb theorized that biological associative memory lies in the synaptic connections between nerve cells, and that the process of learning and memory storage involved changes in the strength with which nerve signals are transmitted across individual synapses. Hebb's Rule states that pairs of neurons that are active simultaneously become stronger by synaptic (weight) changes. The result is a reinforcement of those pathways in the brain. Hebb's Rule states $\Delta w_{ij} = \nu a_i o_j$ where ν is the learning rate that specifies a scaling factor for changes during training.

The Delta Rule, a supervised learning algorithm, additionally states that if there is a difference between the actual output pattern and the desired output pattern during training, then the weights are adjusted to reduce the difference. The Delta Rule states $\Delta w_{ij} = \nu(t_i - a_i)o_j$, where t_i is the training (desired output) pattern. The back-propagation rule is a generalization of the Delta Rule for a network with hidden neurons.

The best learning rule to use with linear neurons is the Delta Rule. This allows arbitrary associations to be learned, provided that the inputs are all linearly independent. Other learning rules (such as Hebb's) require that the inputs also be orthogonal.

The Two Major Topologies

Neural networks can be arbitrarily categorized by topology, neuron model, and training algorithm. (Figure 2 shows one method of classifying neural networks.) There are two main subdivisions of neural network models: Feed-forward and feedback topologies.

Feedback models can be constructed or trained. In a constructed model the

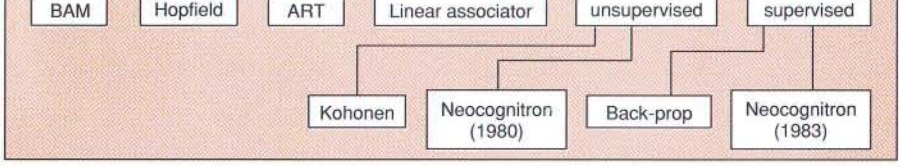


***Figure 2:** A taxonomy of neural-network types*

DBMS Case Study:

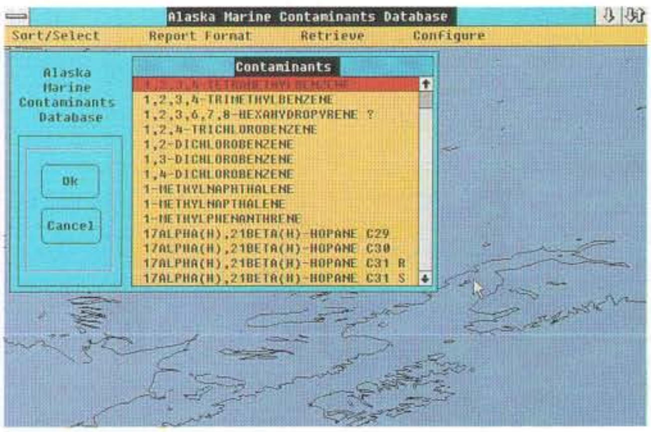
The Exxon Valdez Disaster

March 24, 1989. Exxon VALDEZ tanker runs aground, creating the worst oil spill in U.S. history. 11,000,000 gallons contaminate the pristine waters of Alaska's Prince William Sound.

The Problem Major disasters, like the Exxon Valdez spill, require quick response based on careful data analysis. Fortunately, an easy-to-use database was already being created which would help.

The Application The Alaskan Marine Contaminants Database lets oceanographic chemists easily access 60 megabytes of data covering the past decade. The database is provided free of charge on CD-ROM, and the Windows interface means they can get right to work, assessing damage to the ecosystems of Prince William Sound and other Alaskan waters.

The Solution db_VISTA III is the only DBMS with the features this project required: C language support, Windows compatibility, royalty-free runtime distribution, quick performance in large databases, quality documentation and support. With the Alaskan Marine Contaminants Database, the difficult job of calculating the long-term effects of the Exxon spill is a little easier.*


Alaska Marine Contaminants Database
Sort/Select
Report Format
Retrieve
Configure
Contaminants
Alaska Marine Contaminants Database
Ok
Cancel

```
1,2,3,4-TETRAMETHYLBENZENE
1,2,3,4-TRIMETHYLBENZENE
1,2,3,6,7,8-HEXAHYDROPYRENE ?
1,2,4-TRICHLOROBENZENE
1,2-DICHLOROBENZENE
1,3-DICHLOROBENZENE
1,4-DICHLOROBENZENE
1-METHYLNAPHTHALENE
1-METHYLNAPTHALENE
1-METHYLPHENANTHRENE
17ALPHA(H),21BETA(H)-HOPANE C29
17ALPHA(H),21BETA(H)-HOPANE C30
17ALPHA(H),21BETA(H)-HOPANE C31 R
17ALPHA(H),21BETA(H)-HOPANE C31 S
```

A Microsoft Windows front end lets chemists select regions from a map to retrieve data. And, db_VISTA III's SQL-based query and report writer lets users perform complex SQL data searches.

Your DBMS problems may not make the headlines, but they are no less important and often no less challenging. If you develop applications for MS-DOS, MS Windows, UNIX, VMS, QNX, OS/2, Macintosh, and other environments, db_VISTA III is your solution.

Call 1-800-db-RAIMA (1-800-327-2462)

* Reprints of the story, as published in PC Week and Data Based Advisor, are available from Raima.

db_VISTA III™

Database Management System

Specifications: ***Complete C source code available. No Royalties C Lanaguage Portability & High performance***

Network Data Model. Relational B-tree indexing. Relational SQL query and report writer. Single & Multi-user. Automatic recovery. Built-in referential integrity. Complete revision capability. **Supports:** MS-DOS, MS Windows, UNIX, QNX, SunOS, XENIX, VMS, Macintosh. OS/2 compatible. **Most C Compilers supported. LANs:** 3COM, Novell, Banyan, Appleshare. Call for other environments.

Power Tools For C Programmers

RAIMA™ CORPORATION

db_VISTA III DBMS rated number #1

For Performance and Flexibility of DBMS Programming Tools- **PCWEEK** Poll of Corporate Satisfaction, August 28, 1989.

CIRCLE NO. 287 ON READER SERVICE CARD

DD040W

Raima Corporation 3245 146th Place S.E., Bellevue, WA 98007 USA **(206)747-5570** Telex: 6503018237 MCI UW FAX: (206)747-1991
International Distributors: **U.K.**: (0992) 500919 **Germany**: 07127/5244 **Switzerland**: (01)725 0410 **Netherlands:** (02159)46 814 **Sweden**: (013)124780 **Italy:** 045/584711 **Norway**: (02) 44 88 55 **Denmark**: 2887249 **U.S.S.R.:** (812) 292-1965, (0132) 35-99-08 **Australia:** 02 419 7177 **Japan**: (03)473 7432 **Taiwan**: (02)511 3277 **Mexico**: (83) 57 35 94 **Central America**: 506 28 07 64 **Argentina**: 1 313 5371 **Chile**: 2 696 4308 **Uruguay**: 2 92 0959

(continued from page 40)

weight matrix is created by taking the outer product of every input pattern vector with itself or with an associated input, and adding up all the outer products. After construction, a partial or inaccurate input pattern can be presented to the network, and after a time the network should converge so that one of the original input patterns is the result. Hopfield and BAM are two well-known constructed feedback models.

The Hopfield network is a self-organizing, associative memory. It is the canonical feedback network. It is composed of a single layer of neurons that act as both output and input. The neurons are symmetrically connected ($w_{ij} = w_{ji}$). (See Figure 3.) Hopfield networks are made of nonlinear neurons capable of assuming two output values: -1 (off) and +1 (on). The linear synaptic weights provide global communication of information. In spite of its apparent simplicity, a Hopfield network has considerable computational power.

The weight matrix is created by taking the outer product of each input pattern vector with itself, and adding up all the outer products. After construction, a pattern is given to the network. A process of reaction-stimulation-reaction between neurons occurs until the network settles down into a fixed pattern called a "stable state." Thus, the network result comes as a direct response to input.

The energy required by a device to reach a stable state can be plotted in three dimensions as a curved surface. In this representation, the stable states of the system (the energy minimums) appear as valleys. A neural network, which is used to find "good enough" solutions to optimization problems, may have many possible energy minimums or valleys. Depending upon the initial state of the network, any of the deepest valleys may end up as the answer. Inputing incomplete information to an associative memory network causes the network to follow paths to a nearby energy minimum where the complete information is stored.

Hopfield networks can recognize patterns by matching new inputs with the closest previously stored patterns. Hopfield networks are especially good for finding the best answer out of many possibilities. They are also good at recalling all of a stored piece of information when given partial data. Hopfield networks are often used in applications requiring some form of content addressable memory.

While the Hopfield model is able to associate on a large scale, it does not learn; the weights must be set in advance. A serious limitation of the Hopfield model is that the maximum number of memories M, which can be stored while still retaining perfect recall is [M less than or equal to N/(4 log N)] where N is the number of neurons. If more memories are stored, then the stable states begin to differ significantly from the stored information and eventually all will be forgotten. If an error rate of 5 percent is tolerable, then the capacity is about 14 percent of N. The hardware efficiency is also poor. A variation has been proposed, called the "Unary or Hamming" network, which uses inhibitory lateral connections in the internal neurons. It is claimed that this model has a capacity of M >> N with no errors in the final state.

A neural network learns by adapting to changes in the input

Bart Kosko brought the Hopfield network to its logical conclusion with the BAM. The BAM (bidirectional associative memory) is a generalization of the Hopfield network. Instead of creating the weight matrix with the dot product of a pattern with itself (auto-association), pairs of patterns are used (pair association). After construction of the weight matrix, either pattern can be applied as input to elicit as output the other pattern in the pair.

A trained feedback model is much more complicated because adjustment of the weights affects the signals as they move forward as well as back-

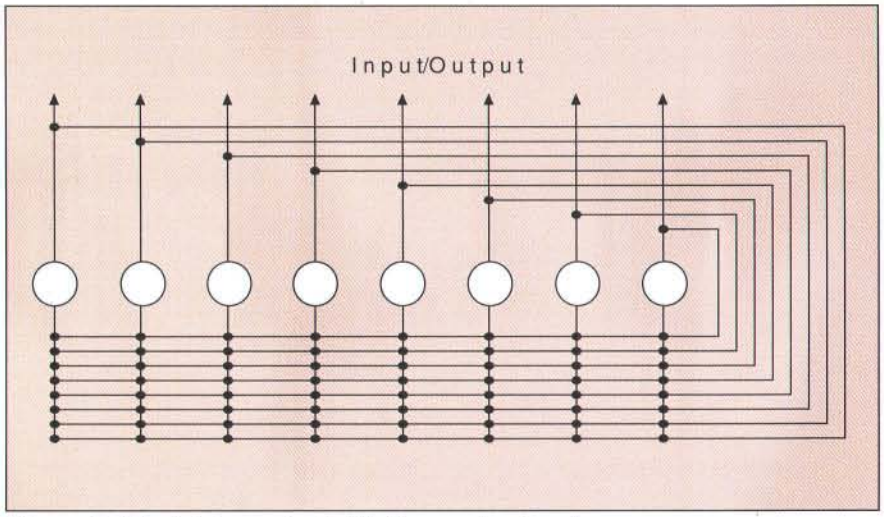


Figure 3: *The topology of a Hopfield neural network*

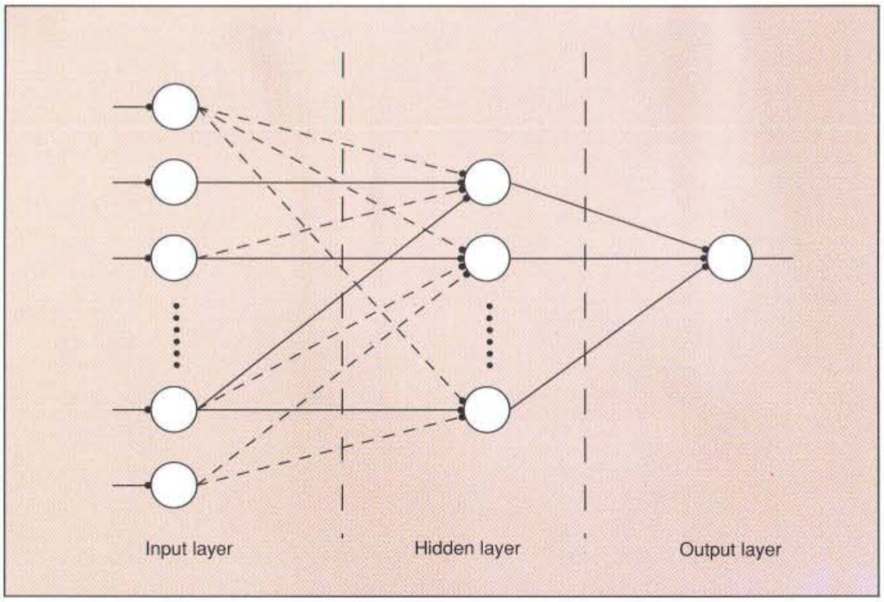


Figure 4: *The topology of a back propagation neural network*

ward. The Adaptive Resonance Theory (ART) model is a complex trained feedback paradigm developed by Stephen Grossberg and Gail Carpenter of the Center for Adaptive Systems at Boston University. ART is considered by some to be very powerful, but the number of patterns that can be stored is limited to exactly the number of nodes in the storage layer. No production applications have been published to date; ART is presently considered a research tool.

Feed-Forward Topologies

The second division of neural networks is the feed-forward category. The earliest neural network models were linear feed-forward. In 1972, two simultaneous papers independently proposed the same model for an associative memory, the linear associator. J.A. Anderson, a neurophysiologist, and Teuvo Kohonen, an electrical engineer, were not aware of each other's work.

The linear associator uses the simple Hebb's Rule. The only case where association is perfect when simple Hebbian learning is used is when the input patterns are orthogonal. This puts an upper limit on the number of patterns that can be stored. The system will work very well for random patterns if the maximum number of patterns to be stored is 10 – 20 percent of the number of neurons. If the input patterns are not orthogonal, there will be interference among them; fewer patterns can be stored and correctly retrieved. One of the predictions of the linear associator is interference between nonorthogonal patterns. Much of Kohonen's book, *Self-Organization and Associative Memory* (Springer-Verlag, 1984) is concerned with correcting the errors caused by interference.

The nonlinear feed-forward models are the most commonly used today. Feed-forward networks, for historical reasons, are less often considered to be associative memories than the feedback networks, even though they can provide exactly the same functionality. It can be shown mathematically that any feedback network has an equivalent feed-forward network that performs the same task.

Types of Learning Algorithms

There are two main types of training algorithms: Supervised and unsupervised. Supervised learning is the most elementary form of adaptation. It requires an a priori knowledge of what the result should be. During training, the network's output is compared to the ideal response, and any error is used to correct the network. Learning occurs as a result of changes to the weights to reduce the errors as the network gains experience. For one-layer networks this is easily accomplished by monitoring each neuron individually. In multi-layer networks, supervised learning is more difficult due to the correction of the hidden layers. Unsupervised learning differs in that it does not have specific corrections made by comparison to ideal results. Supervised and unsupervised learning are methods which are used exclusively of each other.

The supervised back propagation model is the most commonly implemented paradigm today because it is the best general-purpose model and probably the best at generalization. (This model is used by the "BrainMaker" software from California Scientific Software.) Back propagation is a multi-layer feed-forward network that uses the Generalized Delta Rule.

By 1985, back propagation had been simultaneously discovered by three groups of people: D.E. Rumelhart, G.E. Hinton, R.J. Williams; Y. Le Cun; and D. Parker. Back propagation is the canonical feed-forward network where an error signal is fed back through the network, altering weights as it goes, in order to prevent the same error from happening again. (See Figure 4.)

The error on an output neuron, i, for a particular pattern, p, is defined as $E_{pi} = (T_{pi} - O_{pi})$ where T is the training (desired) pattern and O is the actual output. The total error on pattern p, E_p, is the sum of the errors on all the output neurons for pattern p. The total error, E, for all patterns is the sum of the errors on each pattern over all p. The simplest method for finding the minimum of E is known as "gradient descent." It involves moving a small step down the local gradient of the scalar field. This is directly analogous to a skier always moving down hill through the mountains until he hits the bottom.

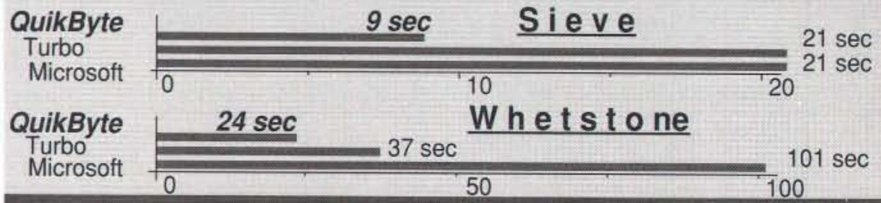
Back propagation is useful because it provides a mathematical explanation for the dynamics of the learning process. It is also very consistent and reliable in the kinds of applications that can currently be built. The biggest limi-

QuikByte PASCAL

The best PASCAL for the PC environment

- **Microsoft object code compatible**
  .OBJ interface to MASM, MS C, BASIC, FORTRAN, COBOL
- **True Pascal - ISO/ANSI standards**
  Switch selectable language extensions
- **Superb math library**
  Unequaled for the PC, in line code generation
- **Ultra smart optimizing linker**
  For smaller .EXE files
- **Turbo-SC** (Turbo Super-Charger)
  Turbo dialect to ***QuikByte*** **PASCAL** translator

and most highly optimizing compiler of any PC-HOL



***QuikByte* Software, Inc.** *Phone (213) 791 2095 Fax (213) 791 2094*

23820 Hawthorne Blvd. Suite 101, Torrance, California 90505 USA

***QuikByte* PASCAL**	***QuikByte* Turbo-SC**
The Professionals Choice	*Source to Source Translator*
Only **149.95**	Only **144.95** Californians add Sales Tax
ISO/ANSI extended	*For twice the Performance*

CIRCLE NO. 427 ON READER SERVICE CARD

tation is the size of the network. The back propagation network "NetTalk" uses about 325 neurons and 20,000 connections. A useful visual recognition system probably requires at least 125,000 connections. Currently available commercial systems provide anywhere from a few neurons and connections to 1 million neurons and 1.5 million connections, for anywhere from $200 to $25,000.

A popular unsupervised feed-forward model is the Kohonen model. The basic system is a one- or two-dimensional array of threshold-type logic units with short-range lateral connections between neighboring neurons. The system modifies itself so that nearby neurons respond similarly. The neurons compete in a modified winner-take-all manner. The neuron whose weight vector generates the largest dot product with the input vector is the winner and is permitted to output. In this model not only the weights of the winner but also those of its nearest neighbors (in the physical sense) are adjusted.

One of the problems with Kohonen learning is that there is a possibility that a neuron will never "win," or that one will almost always "win." The weight vectors get stuck in isolated regions. One way to prevent the weight vectors from getting stuck is to start off with all the weight vectors equal. The network is first fed fractional amounts of the patterns. The inputs are then slowly built up to the full input patterns. This method, called "convex combination," works well but it slows down learning. Another preventative method is to add noise to the data, which makes the probability density function positive everywhere. The probability density function is a real-valued function that gives the probability that a random variable has values in the set. This method works, but it is even slower than convex combination. Another approach is to give the neurons a "conscience"; if the neurons realize that they are winning a lot, they will step out of the competition for a while.

A special case of the feed-forward model is the Neocognitron. The original model was unsupervised, but a more recent model (1983) uses a teacher. The multi-layer (seven- or nine-layer) system assumes that the builder of the network knows roughly what kind of result is wanted. All the neurons are of analog type; the inputs and outputs take nonnegative values proportional to the instantaneous firing frequencies of actual biological neurons. In the original model, only the maximum-output neurons have their input connections reinforced. It uses a variation of the Hebbian Rule. After learning is completed, the final Neocognitron system is capable of recognizing handwritten numerals presented in any visual field location, even with considerable distortion. Drawbacks of the Neocognitron are that it is highly specialized and requires a large number of neurons and connections.

Conclusion

Neural networks are capable of some impressive things but they are also limited, primarily by the size of the network and the complexity of the problem. They are especially good at association and generalization, but poor at precise computations and logic. Some models are able to generalize better than others, some are good at association.

With more than 40 functioning models to choose from, it is important to know which models have had the most success and to understand their similarities and differences. Currently, back propagation is the most popular model. Several others are discussed in detail in this issue, each has it own merits.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 3.**

"Compiler Ads Are Confusing."

They all claim that their products are the fastest and most powerful. Buzz words like optimized, integrated, and modular are everywhere—never meaning quite the same thing.

We'd like to be more direct. We'll tell you what you can do with our compiler—then you make the comparisons.

■ DUAL PERFORMANCE You have two compilers in one integrated package—Quick for speed applications development and optimizing for the best code generation—with a simple menu option to move between the two. ■ FLEXIBILITY You can interface directly with C or any other language. Write only one set of sources for DOS and OS/2, run the most complex applications with no change. ■ COMPATIBILITY You can generate code compatible with Microsoft Windows, using all window facilities. And develop Presentation Manager applications with no additional software. ■ OPTIMIZATION You get true global optimization, using data flow analysis and proprietary techniques, not just the standard peephole optimization and automatic assignment of variables to registers. ■ ENVIRONMENT You have many features you won't find in any other environment—like the ability to organize your code into separate libraries and set compiler options both globally and on a per-module basis. And a make facility that is so well integrated, you don't even know it's there. ■ TOOLS You get a debugger, profiler, object librarian and overlay linker with unique capabilities. And a runtime library with surprises like interrupt driven serial communications, true multitasking, graphics, and mouse interface modules.

Stony Brook Professional Modula-2 (both the Quick and optimizing compilers for DOS *and* OS/2) for $295. Stony Brook QuickMod (for DOS *or* OS/2) for $95.

Stony Brook—we eliminate the confusion.

■ **The fine print version of this information with all the details, including our benchmark performances, will be mailed to you within 24 hours if you call our 800 number.**

800/624-7487 **805/496-5837** California and International

805/496-7429 Fax 187 East Wilbur Road, Suite 9 Thousand Oaks, CA 91360

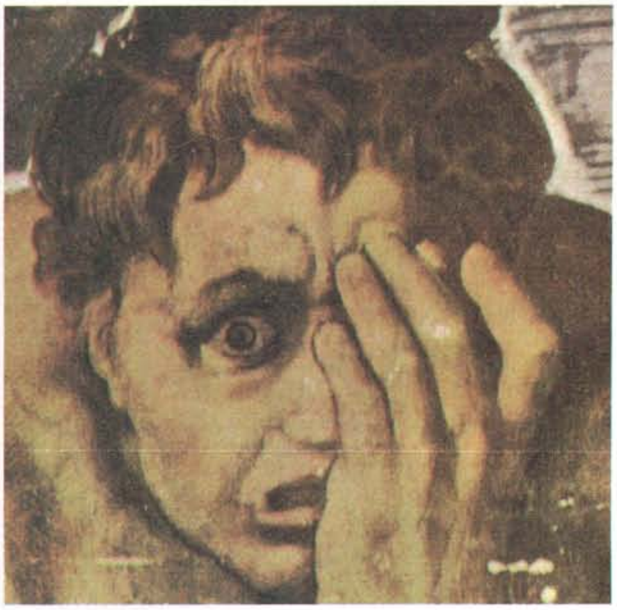
Stony Brook SOFTWARE

Your Partner in Software Development

©1989 Gogesch Micro Systems, Inc.

CIRCLE NO. 232 ON READER SERVICE CARD

Death Taxes Software Piracy



We can save you from one of them.

Sorry. Death we can't do anything about. As for taxes, when you use our product you'll probably wind up paying more. But software piracy: there we offer some help. Our family of software protection devices (dongles) have improved unit sales for over 2,000 companies around the world. Our products can be used in the MS-DOS, OS/2 and Macintosh environments.

Build Your Own Custom Protection Environment

Use our patented "dual-locking" ASIC chip as the basic building platform. Next, add options like: on-the-fly read/write memory, write-once or multiple-write locking codes, and encryption shells. Then add your own programming creativity to build a protection environment best suited to your product.

Users attach the device to their parallel port, and programs won't run without it. Back-up copies, hard disk and LAN operation are not interfered with.

Your Intellectual Property Belongs To You

And if you don't protect it, who will? Our products offer the most equitable way to protect your interests without sacrificing the rights of your customers. Call us today for information and demonstration units.

Software Security™

1011 High Ridge Road - Stamford, CT 06905

1-800-333-0407 ext. 102

203-329-8870 Fax 203-329-7428 BBS 203-329-7253

AppleLink™ D2379

Macintosh is a trade mark of Apple Computer Inc., Activator, Mactivator are trade marks of Software Security, Inc. illustration: detail from Michelangelo's Last Judgement

CIRCLE NO. 423 ON READER SERVICE CARD

Implementing the Rhealstone Real-Time Benchmark

Where a proposal's rubber meets the real-time road

Rabindra P. Kar

In February 1989, the late Kent Porter and I proposed a set of benchmarking operations for real-time multitasking systems (see "Rhealstone: A Real-Time Benchmarking Proposal," *DDJ*, February 1989). That article generated a lot of interest and many valuable suggestions from *DDJ* readers, as well as others in the real-time software community. The reader response made it possible for us to refine and clarify several important aspects of the original benchmark proposal. I am very grateful to those of you who shared your insights with us.

This article presents the refined definition of the Rhealstone benchmark. It also contains a suite of C programs that implement the benchmark under iRMX, a real-time operating system from Intel. Refer to the original proposal for the rationale behind proposing Rhealstones in the first place, and for background information on the real-time multitasking operations that comprise it.

First, I'll give a quick summary of what the Rhealstone benchmark is and what it seeks to measure. The benchmark identifies the execution times (or time delays) associated with six operations that are vital indicators of real-time multitasking system performance. These six operations are named "Rhealstone components." When a real-time system is benchmarked, each of these components is measured separately. The empirical results are combined into a single figure of merit (Rhealstones per time unit). There are two ways of calculating the Rhealstone number: One of them generic and one of them appropriately weighted for a particular type of application (an application-specific Rhealstone).

Rhealstones are intended to provide a standard of comparison between real-time computers across the industry. Hence, their specification is: a. Independent of the features found in any CPU; b. Independent of any computer bus architecture; c. Independent of the features or primitives of any operating system or kernel (collectively referred to hereafter as real-time executives).

The C language implementation of the benchmarks is, of course, specific to an operating system (iRMX in this case). However, the OS-specific part of the code is confined to a few system

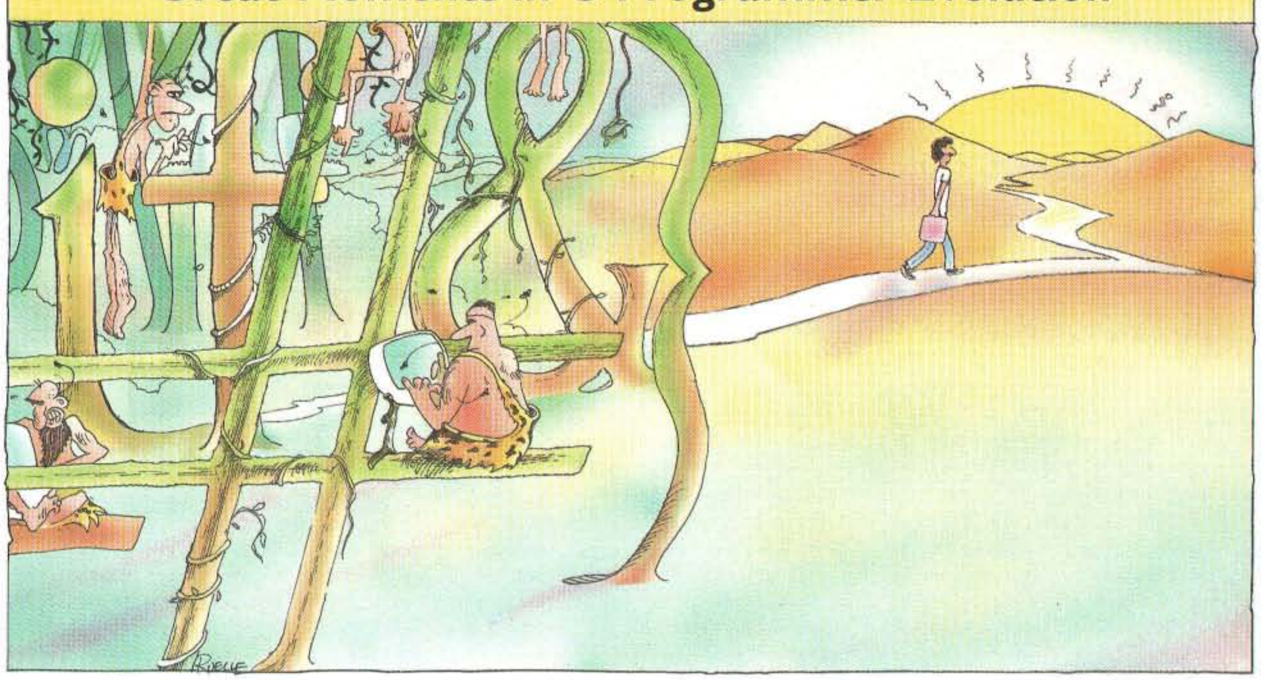
Robin is a senior engineer with the Intel Systems Group and can be reached at 5200 N.E. Elam Young Parkway, Hillsboro, OR 97124-6497.

Reader's Rhealstone Recommendations

When the original Rhealstone proposal was put forth more than a year ago, *DDJ* solicited comments, suggestions, and recommendations from readers. In addition to the individuals Robin has acknowledged in this article, the following readers contributed comments. If any contributors were left off this list, it is unintentional and we apologize — please let us know who you are. The version of the benchmark presented in this article does not necessarily represent Rhealstone as it will look in years to come. We look forward to your suggestions and recommendations for this version too.
— Eds.

Mark Smotherman, Clemson University; Glenn Yeager, Applied Integration Management Corp.; Colburn L. Norton, Baytown, Texas; Gary Osborne, Apricot Computers; Jim D. Hart, Papillion, Nebraska; John Morgan, Bellingham, Washinton; G. Bruce Lott, Real Time Systems; Michael S. Sossi, Leo Burnett USA; Rudi Borth, Stratford, Ontario; Carol Sigda, Industrial Programming Inc.; Phil Daley; Hillsboro, New Hampshire; Tim Olson, Advanced Micro Devices.

Great Moments in C-Programmer Evolution



Code-dweller emerges from the jungle

"It's a jungle in there," said the programmer looking at the code for the user interface of an application. "Every year it gets worse."

Don't despair. Finally, there is a way out. Vermont Views™ 2.0.

From Complexity to Simplicity

Vermont Views 2.0 replaces the complexities of interface coding with the simplicity of the **Vermont Views Designer**. This powerful interactive forms designer works in concert with our comprehensive library of over 550 functions to make interface development and management quicker and easier than ever before.

Development Will Never Be the Same Again

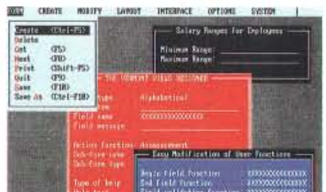
With the Vermont Views Designer you will quickly create operational prototypes of an application interface—and enjoy doing it! Because design is fast and visual, you will involve your clients actively from the beginning. Last-minute change requests will be accepted without battles or escalating costs.

No longer will you throw away months of prototype code—the prototype will become the implementation. And, integration and final testing will go faster, because all Designer objects are tested for validity as they are created.

No More Maintenance Blues

Software maintenance typically accounts for over 50 percent of total lifecycle programming effort—and a higher percentage of headaches. With the Vermont Views Designer, you will always be able to revise the interface quickly and easily, seeing the changes as you make them.

The Vermont Views Difference



Editing form. Enter text. <Alt-J> jumps to menu. <F1> for help.

Screen generators for most C libraries require you to modify generated source code to create fully functional forms—after which you can no longer use the screen generator. Not so with the Vermont Views Designer. Designer forms and menus can incorporate any of the special capabilities of Vermont Views—such as nested menus, scroll bars, tickertape fields, scrollable form regions, choice lists, and memo fields—and still be revised interactively.

Message from the Jungle

"At a recent field staff meeting, we were able to get a consensus on what forms should look like by using the Designer on a big screen TV. Changes can be posted real-time, and a functioning prototype results from the exercise. The form designer is GREAT."

—Randy Jones, Beta Tester

Globally Applicable

Use Vermont Views with any database or file manager with a C-language interface, such as Oracle, Informix, dBase, Clipper, dbVista, Btrieve, and C-tree. Maintain the same interface with the same source code under DOS, OS/2, UNIX, XENIX, and VMS.

Create interfaces for any roman-based language. Truly a global solution for your interface needs.

100% No-Risk Guarantee

We believe in our product. Try Vermont Views for as long as you want. No limits. If not fully satisfied, return for a full refund.

Free Test Drive

Call now for a free DOS working copy of the Designer, lacking only the ability to save forms. Get out of that jungle!

Call **800-848-1248**
Fax **802-848-3502**

Vermont
Creative
Software

Pinnacle Meadows, Richford, VT 05476
Phone: 802-848-7731 Telex: 510-601-4160

(continued from page 46)

calls (to create tasks, put them to sleep, accept/relinquish semaphores, and so on) that are found in almost every multitasking executive. The C benchmark source is easily portable to most other executives if the iRMX system calls in it are replaced by the equivalent system calls of the target executive.

When a computer is used in a real-time-control situation, the solution usually fits the model described in Figure 1.

"Real-time computer system" refers jointly to the CPU and system software (OS, kernel, or combination thereof) that provide a base execution vehicle for the application software. The Rhealstone benchmark helps the real-time solution designer choose the highest-performance real-time computer available as the execution base for the project. Rhealstones are not designed to measure how good the complete solution is, and they may not be an appropriate measure for the end user.

Rhealstone Components

The specifications for the six real-time operations (Rhealstone components) that comprise the benchmark are detailed in the following section. For a graphical specification of each component, refer to Figures 2 through 7. Each component's specification is followed by a paragraph (or two) describing the C benchmark program used to measure the Rhealstone component on iRMX II. It is important to realize that the verbal and graphical specifications, not the C programs, are the essential core of the benchmark. It is entirely possible to obtain more accurate Rhealstone component values by using different algorithms or programming languages or special performance-analysis hardware.

The task-switch time (see Figure 2) is the average time to switch between two active tasks of equal priority. The tasks should be independent — that is, there should not be contention between them for hardware resources, semaphores, and so on. Task switching must be achieved synchronously (without preemption) — for example, when the running task puts itself to sleep or when the executive implements a round-robin scheduling algorithm for equal-priority tasks.

Listing One (tswit.c), page 100, is a program to measure task-switch time. The code of *task1* and *task2* is identical and very simple: A loop in which *rqsleep* gets called in every iteration. The iRMX system call *rqsleep(sleep_time, return_code_pointer)* lets a task put itself to sleep (suspend execution) for *sleep_time* system clock ticks (in iRMX, the default clock tick is 10 milliseconds long). If *sleep_time* is 0, the task is not necessarily put to sleep; rather, iRMX will switch execution to any other equal-priority task that is ready to run (which is why this program calls *rqsleep*).

The *return_code_pointer* is the last parameter of every iRMX system call. It points to an unsigned variable in which iRMX places the status of the call. If the status returned is 0, the call has been executed correctly; if this is not so, the status returned is an error or warning code indicating why the call did not execute as it should have.

The call *rqgettime (return_code_pointer)* returns the number of seconds that have elapsed since a fixed point in time. It is called twice, to measure the elapsed time (in seconds) between any two points in a program.

The call *rqcreatetask (priority_level, start_address, data_seg, stack_pointer, stack_size, task_flags, return_code_pointer)* creates a new task, as its name suggests. The first parameter sets the new task's priority between 0 (highest priority) and 255 (lowest priority). The other parameters are either self-explanatory or iRMX programming details. This call returns a *task_token* that becomes iRMX's identifier for the task. Consequently, the *rqdeletetask(task_token, return_code_pointer)* call uses *task_token* to identify which task is to be deleted. If *task_token* is NULL or 0, the task deletes itself.

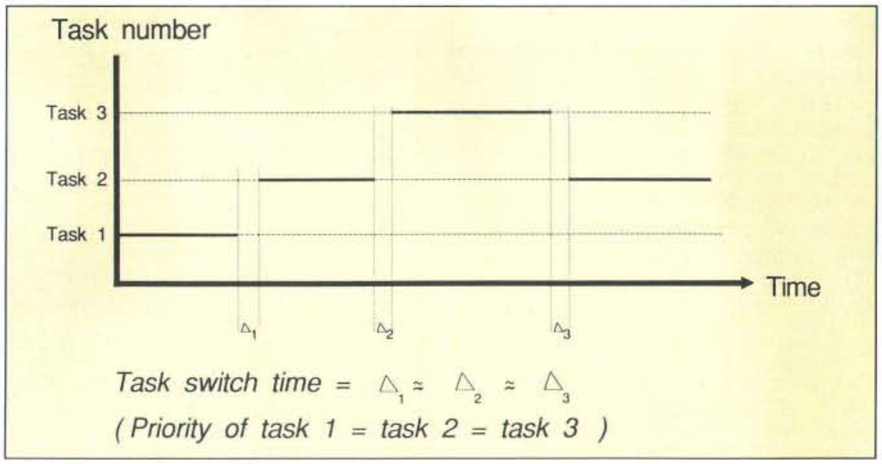
The *rqgetpriority(task_token, return_code_pointer)* call lets a task find out the priority level of any existing task in the system. If *task_token* is NULL or 0, the priority of the calling task itself is returned.

Finally, the call *rqsetpriority(task_token, priority_level, return_code_pointer)* lets a task dynamically change its own or any other task's priority. I've used it to set the main task's priority to be lower than those of *task1* and *task2* so those tasks can run to completion without interference from the main program.

The preemption time (see Figure 3) is the average time for a high-priority task to preempt a running low-priority

Real-time computer system	+	Real-time application code	+	Human-machine and/or machine-machine interface(s)	=	Real-time solution

Figure 1: *Typical real-time control model*



Task number
Task 3
Task 2
Task 1
Time
Δ1
Δ2
Δ3
Task switch time = Δ1 ≈ Δ2 ≈ Δ3
(Priority of task 1 = task 2 = task 3)

Figure 2: *Task-switch time*

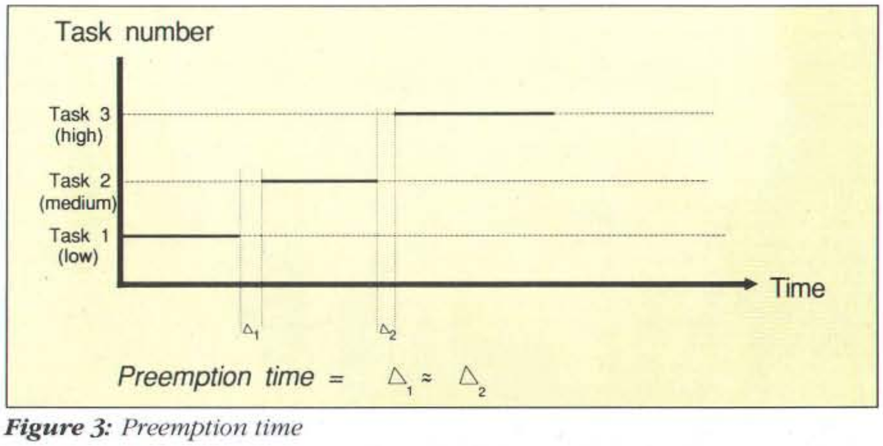
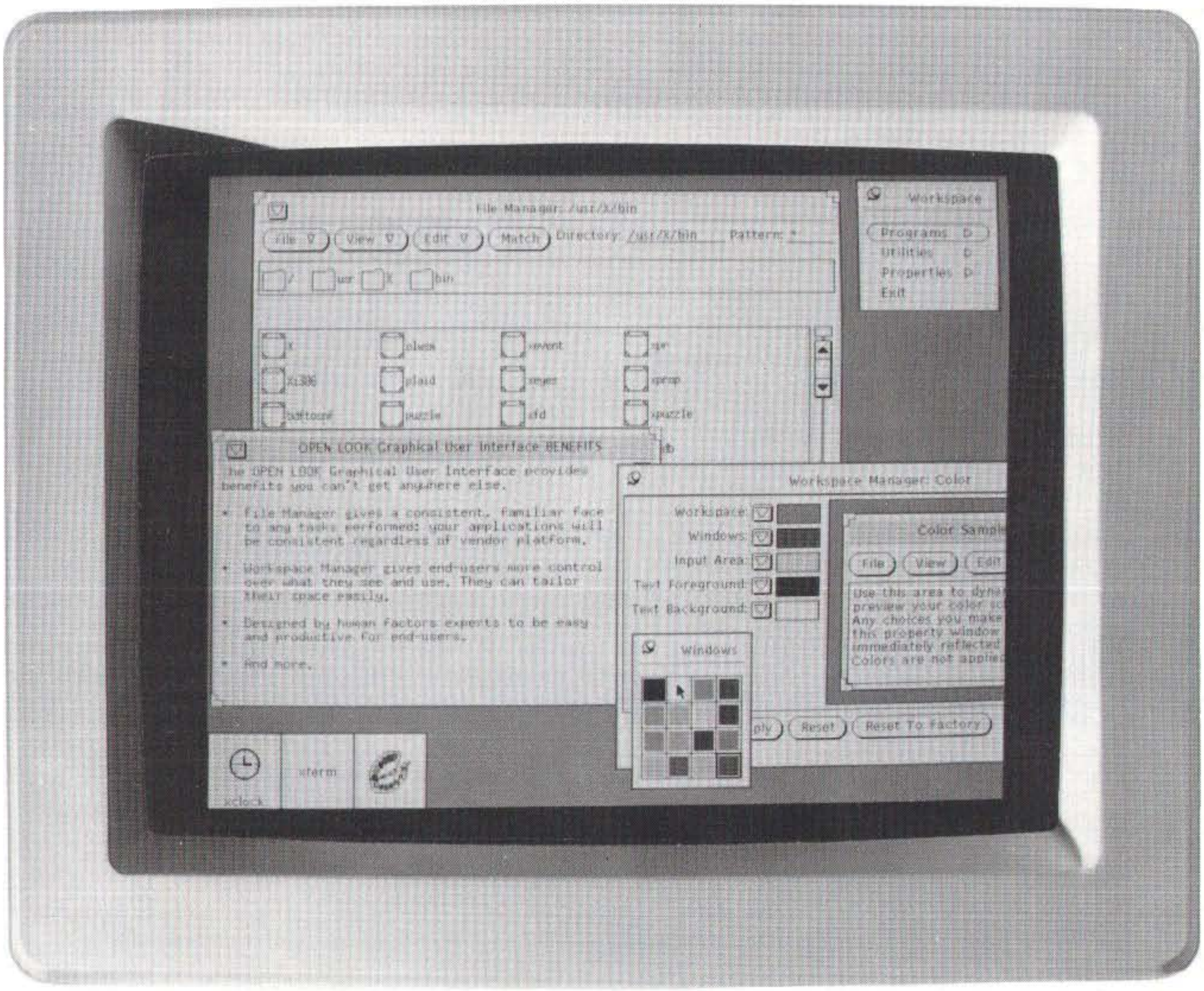


Figure 3: *Preemption time*

UNIX® System V, In The Easy Open Package.



OPEN LOOK Graphical User Interface BENEFITS

The OPEN LOOK Graphical User Interface provides benefits you can't get anywhere else.

- File Manager gives a consistent, familiar face to any tasks performed: your applications will be consistent regardless of vendor platform.
- Workspace Manager gives end-users more control over what they see and use. They can tailor their space easily.
- Designed by human factors experts to be easy and productive for end-users.
- And more.

Use this area to dyna
preview your color sc
Any choices you make
this property window
immediately reflected
Colors are not applied

The AT&T OPEN LOOK™ Graphical User Interface—the standard look and feel for UNIX® System V Release 4—is now shipped on every source tape. Human factors research drove its design, so wizards and novices alike learn it more easily than competing products.

Key software developers and leading computer makers (like those listed below) already support it and are committed to shipping products.

In addition, the AT&T OPEN LOOK X Toolkit is 100% compatible with MIT Intrinsics and with ICCCM's rules for interoperability. Best of all, our complete specification and unique style guide help you program more easily today, and plan for tomorrow.

The AT&T OPEN LOOK Interface can give you easy, open access to a wealth of new business opportunities. For source code licensing information and a technical prospectus, call 1-800-828-UNIX, ext. 537. International customers may also contact us in Tokyo at +81-3-431-3670, or London at +44-1-567-7711.

Software For The Open-Minded.

AT&T
The right choice.

ALTOS INDIA LTD. • AMDAHL CORPORATION • ARIX CORPORATION • BRISTOL GROUP • COMMODORE BUSINESS MACHINES, INC. • ELAN COMPUTER GROUP, INC. • EXOC • FICOR • FUJITSU, LTD. • GRAPHIC SOFTWARE SYSTEMS, INC. • HARRIS CORPORATION • HCL LTD. • ICL • INFORMIX • INGRES • INTEL • INTERACTIVE SYSTEMS CORP. • KONSULTHUSETDATA • MOTOROLA • OLIVETTI • ORACLE • PYRAMID • QUANTUM SOFTWARE SYSTEMS LTD. • QUEST SYSTEMS CORPORATION • SOLBOURNE COMPUTER, INC. • SUN MICROSYSTEMS INC. • SYBASE • SYSTEM STRATEGIES LTD. • TOLERANT SOFTWARE • TOSHIBA CORPORATION • UNIFY CORPORATION • VERSASOFT CORPORATION • VISUAL EDGE SOFTWARE LTD. • V SYSTEMS, INC. • WIPRO INFORMATION TECHNOLOGY LTD.

INTROL

CROSS DEVELOPMENT SYSTEMS

- INTROL-C Cross-Compilers
- INTROL-Modula-2 Cross-Compilers
- INTROL-Macro Cross-Assemblers

Provide cost and time efficiency in development and debugging of embedded microprocessor systems

All compiler systems include: Compiler • Cross-assembler • Support utilities • Runtime library, including multi-tasking executive • Linker • One year maintenance • User's manual, etc.

TARGETS SUPPORTED:
6301/03 • 6801/03 • 6804 • 6805 • 6809 • 68HC11 • 68000/08/10/12 • 32000/32/81/82 • 68020/030/881/851

AVAILABLE FOR FOLLOWING HOSTS:
VAX & MicroVAX; Apollo; SUN; Hewlett-Packard; Gould PowerNode; Macintosh; IBM-PC, XT, AT and compatibles

INTROL CROSS-DEVELOPMENT SYSTEMS are proven, accepted, and will save you time, money, effort with your development. All INTROL products are backed by full technical support. CALL or WRITE for facts NOW:

INTROL CORPORATION

9220 W. Howard Ave., Milwaukee, WI 53228
414/327-7171 FAX: 414/327-7734
Quality Software Since 1979

CIRCLE NO. 303 ON READER SERVICE CARD

(continued from page 48)
task. Preemption usually occurs when the high-priority task goes from a suspended or sleeping state to a ready state because either the high-priority task wakes up from a previously initiated sleep, or some event or signal that the task was waiting for is recognized. The first case will likely yield a lower preemption time on most systems, and that value is acceptable for the Rhealstone benchmark.

Listing Two (preempt.c), page 100, measures preemption time in an iRMX II system. *task1* (lower priority) merely sits in a delay loop, waiting to be preempted. *task2* loops on an *rqsleep* call. Every time it calls *rqsleep* a (nonpreemptive) switch to *task1* takes place. When one sleep period is over, *task2* wakes up and preempts *task1.*

Interrupt latency (see Figure 4) is the average delay between the CPU's receipt of an interrupt request and the execution of the first application-specific instruction in an interrupt-service routine. The time required to execute machine instructions that save the CPU's context (CPU's and coprocessor's data registers, mode registers, and so on) are part of the interrupt latency.

As just defined, interrupt latency reflects only the delay introduced by the executive and the CPU itself. It does not include delays that occur on the system bus or electrical delays in control circuitry external to the computer system because control circuitry is usually specific to the application.

Interrupt latency can be measured under iRMX II on a PC/AT-compatible computer using ltncy.c (Listing Three, page 100) and latch.asm (Listing Four, page 101). The benchmarking technique used here is different from the one I've used for the other Rhealstone components because interrupt latency is about an order of magnitude smaller than those components (under iRMX), hence the need for greater accuracy in measurement. The first difference is that this benchmark involves a C and an assembler program. Secondly, the latency is measured by reading the 8254 timer chip directly (bypassing iRMX). Of course, this makes the benchmark hardware-dependent, which is the major disadvantage of this technique.

The code in ltncy.c sets up a new interrupt vector (pointing to the assembler code in latch.asm) with the *rqsetinterrupt(encoded_int_level, int_task_flag, int_handler, int_handler_data_seg, return_code_pointer)* call, reads the timer chip, and simulates a hardware interrupt with an *INT (causeinterrupt)* instruction in software. The processor vectors to latch.asm, which saves context (by pushing CPU registers on the stack) and reads the timer again.

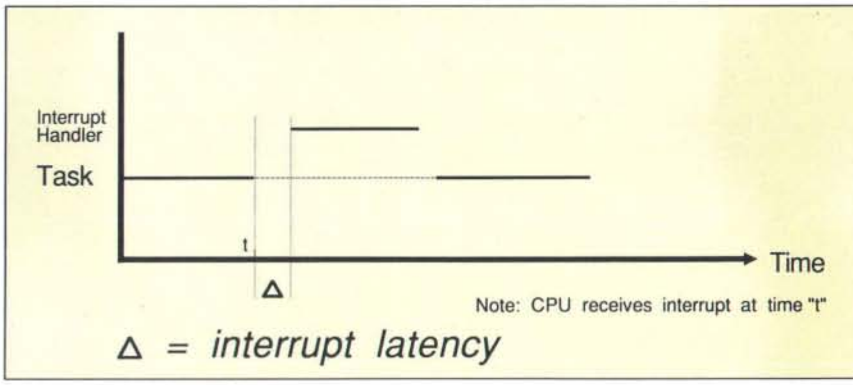


***Figure 4:** Interrupt latency*

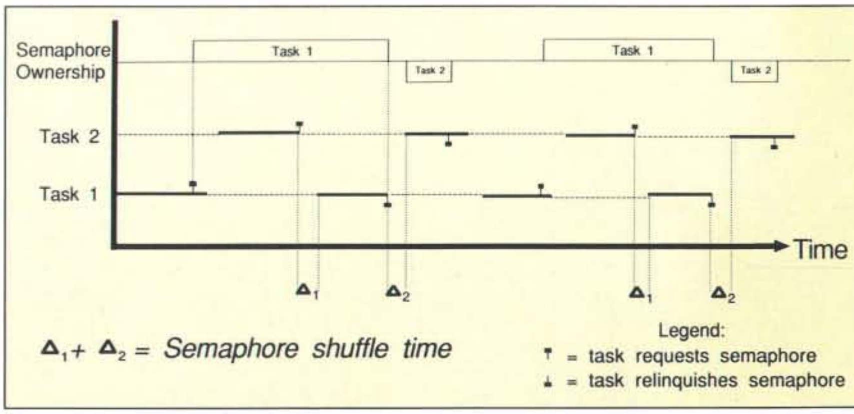

Semaphore Ownership
Task 1
Task 2
Task 1
Task 2
Task 2
Task 1
Time
Δ1
Δ2
Δ1
Δ2
Δ1 + Δ2 = Semaphore shuffle time
Legend:
= task requests semaphore
= task relinquishes semaphore

***Figure 5:** Semaphore-shuffle time*

The difference in the two timer count values is used to calculate interrupt latency in microseconds.

The call *rqresetinterrupt(encoded_ interrupt_level, return_code_ pointer)* merely restores the interrupt vector to its default value (before the *rqsetinterrupt* call).

Semaphore-shuffle time is the delay/overhead, within the executive, before a task acquires a semaphore that is in the possession of another task when the acquisition request is made. Figure 5 illustrates what is being measured. *task2* requests a semaphore that *task1* owns. Semaphore-shuffle time is the delay within the executive (excluding the run time of *task1* before it relinquishes the semaphore) between *task2*s request and its receipt of the semaphore.

The objective here is to measure overhead when a semaphore is used to implement mutual exclusion. Many real-time applications involve multiple tasks needing access to the same resource. Semaphore-based mutual exclusion is a convenient way to ensure that the resource is not interrupted by a second task before it finishes an operation started by the first task.

Listing Five (semshuf.c, page 101) shows a program that measures semaphore-shuffle time. *task1* and *task2* are first executed a fixed number of times, without any semaphore-related calls. Then these tasks are executed again, the same number of times, with a semaphore being shuffled between them at every iteration. The difference in execution time with and without semaphore shuffling is the overhead within the executive. This program uses the following three iRMX system calls associated with semaphores.

The Rhealstone benchmark identifies the execution times (or time delays) associated with six operations that are vital indicators of real-time multitasking system performance

The call *sem_token = rqcreatesemaphore(initial_value, max_value, queuing_method, return_code_ pointer)* creates a new counting semaphore. A counting semaphore can be incremented up to the *max_value* parameter. Because this program sets *max_value* to 1, it is using a simple binary semaphore. The parameter *queuing_method* is set to 0 for a FIFO queuing scheme when more than one task is waiting on the semaphore; a value of 1 would make it a priority-based queue.

The *rqsendunits(sem_token, units_ sent, return_code_ pointer)* call increments the value of the semaphore. In this program, the calling task uses it to relinquish the semaphore.

Finally, the call *remaining_units = rqreceiveunits(sem_token, units_requested, max_wait_time, return_code_ pointer)* decrements the value of the semaphore. The task makes this call to acquire the semaphore if available. The *max_wait_time* parameter specifies (in system clock ticks) the period it is willing to wait for the semaphore. A value of 0xffff means "wait forever."

Deadlock-break time is the average time to break a deadlock caused when a high-priority task preempts a low-priority task that is holding a resource the high-priority task needs.

Figure 6 illustrates how a deadlock

LOGITECH SOFTWARE

Finally. OS/2 debugging help has arrived.

Introducing MultiScope,™ an OS/2, multi-language, CodeView™-compatible debugger, like nothing you've seen before.

For a start, it provides 13 views of your program: Source—Data—Graphic Data—Assembly—Thread—Breakpoint—Watch—Register—Module—Call—Memory—Log—and Output.

Other unique features include: run-time and post-mortem debugging options; OS/2 text mode and Presentation Manager user interfaces; and even a graphical representation of your data structures!

*Call toll free **800-231-7717.** In CA 800-552-8885. Fax 415-792-8901.*

MULTISCOPE™ DEBUGGER

From LOGITECH Software

CIRCLE NO. 131 ON READER SERVICE CARD

Even This Is More Confining Than Clipper.

Just as the vast expanse of the American West gave its settlers a new perspective on opportunity, Clipper's open architecture lends unprecedented freedom to application development.

Unlike fixed systems, Clipper never forces you to "make do". Its language is fully extensible with user-defined functions and new user-defined commands. You can extend the language with routines written in Clipper itself, or integrate code from other languages like C, Assembler, dBASE® and Pascal. Odds are, you already have knowledge you can use with Clipper!

But if a customizable language isn't enough, there's even more elbow room. Database and I/O drivers can be supplemented or replaced. Even Clipper's linker knocks down barriers by allowing you to develop applications larger than available memory, without defining overlays! And when you're done, Clipper's compiler generates stand-alone, executable files for cost-free, unrestricted distribution.

So, don't let the bounds of fixed systems fence you in. Unleash your imagination in the wide-open spaces of Clipper. To find out more, give us a call today.

Clipper® 5.0

The Application Development Standard

213/390-7923

nantucket®

CIRCLE NO. 171 ON READER SERVICE CARD

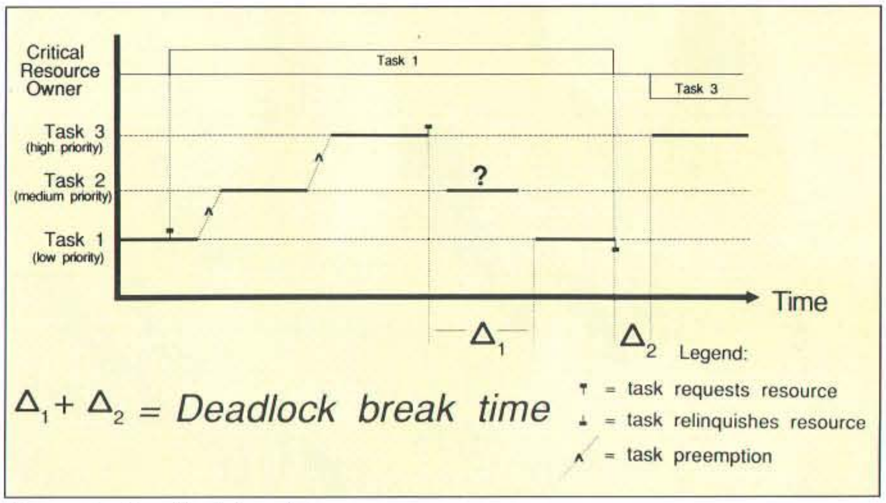
(continued from page 51)
situation occurs. Task 1 gains control of the resource and is preempted by medium-priority task 2. High-priority task 3 preempts task 2 and requests the resource from the executive at some point. Because task 1 still holds the resource, task 3 is now blocked. At this point, an unsophisticated executive would resume task 2. Task 2 knows nothing about the critical resource and may block higher-priority task 1 indefinitely (a deadlock situation!). A good real-time executive would not resume task 2 here (below the ? in Figure 6). To avoid the deadlock, the executive might temporarily raise task 1's priority to the same level as task 3's, until it relinquishes the resource. In any case, the benchmark measures the delay between task 3's request and its acquisition of the resource, excluding task 1's run time before it relinquishes the resource.

The program deadbrk.c (Listing Six, page 102) measures deadlock-break time in iRMX II. The algorithm is similar to the semaphore-shuffle benchmark. Three tasks of different priorities are executed a fixed number of times without competing for a critical resource. The same tasks are executed again, but this time *task1* and *task3* both access the same resource, with a potential deadlock situation occurring in each iteration. The difference in total execution time between the two cases is a measure of the deadlock break time.

Access to a critical resource is guarded by an iRMX object called a "region." The region is created by the *region_token = rqcreateregion(queuing_method, return_code_pointer)* system call. The *queuing_method* parameter has the same function here as in the *rqcreatesemaphore* call. Because only one task should access the critical resource at a time (mutually exclusive access), each task waits at the *rqreceivecontrol(region_token, return_code_pointer)* call until the region is free. The task must relinquish control with the *rqsendcontrol (return_code_pointer)* call when it has finished with the resource.

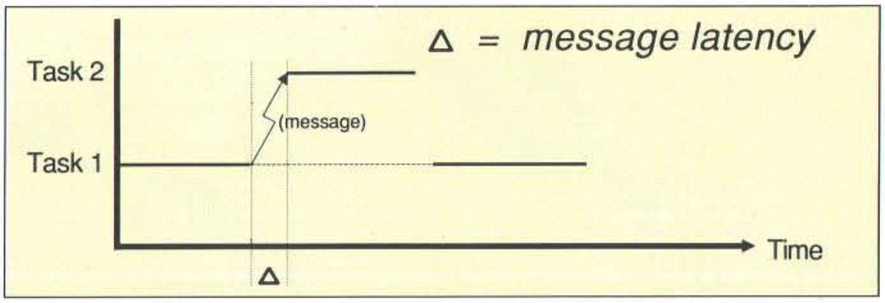
Intertask message latency is the latency/delay within the executive when a nonzero-length data message is sent from one task to another (see Figure 7). To best measure intertask message latency, the sending task should stop executing immediately after sending the message and the receiving task should be suspended while waiting for it.

The message-passing mechanism must obey two important conditions: First, the intertask message-passing link must be established at run time. (Passing data in a predefined memory area,



$\Delta_1 + \Delta_2$ = *Deadlock break time*

Figure 6: *Deadlock-break time*



Δ = *message latency*

Figure 7: *Intertask message latency*

FULL AT&T C++: ANNOUNCING VERSION ~~1.2~~ 2.0!

Guidelines announces its port of **version 2.0** of AT&T's C++ translator. As an object-oriented language, C++ includes: classes, **multiple inheritance**, member functions, constructors and destructors, data hiding, and data abstraction. Object-oriented means that C++ code is more readable, more reliable, and more reusable. And that means faster development, easier maintenance, and the ability to handle more complex projects. C++ is **Bell Labs' answer to Ada and Modula 2.** C++ will more than pay for itself in saved development time on your next project.

C++

from GUIDELINES for the IBM PC: $395

Requires IBM PC-AT or compatible with 512K plus 384K extended memory.
Note: C++ is a *translator*, and requires the use of Microsoft C 4.0 or later.

Here is what you get:

- The full AT&T v2.0 C++ translator with extended memory support.
- Libraries for stream I/O and complex math.
- C++ **Primer**, the definitive book on C++ version 2.0 by **Stanley B. Lippman.**
- Sample programs written in C++.
- Printed installation guide and documentation.
- **30-day money-back guarantee.**

NOW AVAILABLE FOR UNIX V/386 - $495

To Order:

Send check or purchase order to:

GUIDELINES SOFTWARE, INC.
P.O. Box 6368, Dept. DDJ
Moraga, CA 94570

To order with VISA or MC, phone (415) 376-5527. (California residents add sales tax.)

C++ was ported by GUIDELINES under license from AT&T.
Call or write for a free C++ information package.

CIRCLE NO. 351 ON READER SERVICE CARD

such as a global variable, is not permitted.) Second, if multiple messages are sent on the same link, the sending task must not be allowed to overwrite an old message with a new one before the receiving task gets a chance to read it. Multitasking executives typically offer mechanisms such as pipes, queues, and stream files for intertask data communications.

The program it_msg.c (Listing Seven, page 104) measures message latency in iRMX II. Data messages are passed between tasks in an iRMX mailbox. The mailbox is created by the *mailbox_token = rqcreatemailbox(type_flags, return_ cod_ pointer)* system call. A task sends a data message to another task by calling *rqsenddata(mailbox_token, message_ pointer, message_length, return_code_ pointer).*

The receiving task calls *message_length = rqreceivedata(mailbox_token, receive_buffer, max_wait_time, return_code_ pointer)* to receive the message, if available. If not, the task is made to wait until *max_wait_time* clock ticks have elapsed (if this parameter is 0xffff, the receiving task is willing to wait as long as is necessary).

Note: This Rhealstone component is a modification of "datagram throughput," which was proposed in the original article. The specification of intertask message latency partly reflects reader input (see acknowledgment 1).

Computing a Rhealstone Performance Number

Measurement of all six Rhealstone components yields a set of time values (in the tens of microseconds to milliseconds range, for most PCs). Although the individual measurements are of significance by themselves, it is useful to combine them into a single real-time figure of merit, so overall comparisons between real-time computers can be made. To get a single Rhealstone performance number, the following computational steps are necessary:

1. All Rhealstone component time values should be expressed in the same unit (seconds).
2. The arithmetic mean of the components must be computed.
3. The mean (from step 2) must be arithmetically inverted to obtain a consolidated real-time figure of merit, in Rhealstones/second.

Given the following set of measured values:

task-switch time = t1 seconds
preemption time = t2 seconds
interrupt latency = t3 seconds
semaphore-shuffle time = t4 seconds
deadlock-break time = t5 seconds
intertask message latency = t6 seconds

the arithmetic average of the Rhealstone components is:

$$t' = (t1 + t2 + t3 \ldots\ldots + t6) / 6$$

and the system's consolidated real-time performance number is:

1/t' Rhealstones/second

Application-Specific Rhealstones

The operational definition of Rhealstones, in the previous section, is generic for any application that may be executed on a real-time computer. It treats all the Rhealstone components as equally important parameters of real-time performance. Generic benchmarks are useful when evaluating real-time system performance without a particular application in mind.

When a real-time computer is "dedicated" to a type of application, the Rhealstone figure can be computed in a way that is appropriate to it. This performance figure is called an "application-specific Rhealstone." It gives unequal weight to different Rhealstone components because the application's performance is not influenced by all

NOW ON NEW PLATFORMS

VAX/VMS
HP 9000/300
XENIX/386
Motorola UNIX V.3
Sun-3
NCR Tower

OREGON C++
- A true compiler – not a translator
- Three compilers in one: C++, ANSI C or K&R C
- Source-level debugger
- Conforms to AT&T V2.0 C++

C++ SOFTWARE ENGINEERING SEMINARS

Motorola UNIX V.3
VAX/VMS & ULTRIX
Sun-3, 386i
XENIX/386

PASCAL-2
- Full ISO Level 1 standard
- Comprehensive error checking
- Access to command-line arguments
- Extensions include full string type, separate compilations, external variables & more

XENIX/386
Sun-3
VAX/VMS

OREGON MODULA-2
- Conforms to emerging ISO standard
- Excellent support of coroutine and interrupts
- Ideal for embedded systems

680X0 CROSS-COMPILERS ALSO AVAILABLE

CALL 1-800-874-8501
(503) 245-2202 FAX: (503) 245-8449

OREGON SOFTWARE

6915 SW Macadam, Suite 200, Portland, OR 97219 USA

The following are trademarks: Oregon Software, Oregon C++, Oregon Modula-2, Pascal-2, Oregon Software, Inc., Xenix, Sun-3, 386i, VAX, VMS, Motorola, UNIX, AT&T, ULTRIX, NCR Tower, HP 9000

80x86 ROM Development
for Professional Programmers

C_thru_ROM is a complete ROM development package with all the necessary tools to produce a full standalone embedded system using Microsoft-C or Turbo-C.

The C_thru_ROM package includes:

- Comprehensive debugger
- Remote debugging cabability
- Ready-to-use startup code.
- Full 80x86 locator
- ROMable library
- BBS and phone support

C_thru_ROM — $495 (30-Day Money Back Guarantee)

ROM-DOS is Datalight's new ROMable operating system. It goes light on memory usage and is very affordable:

- Uses only 29K ROM and as little as 6K RAM
- Functionality of DOS 3.2 less networking
- Runs your EXE or standard PC programs
- Loads easily into ROM and Boots from ROM or disk
- Supports AUTOEXEC.BAT and CONFIG.SYS
- Full device driver support
- Only $6 per copy in quantity
- Developers Kit has code, tools, and utilities to install, port, create ROM images, etc.

Free demo disk lets you run your application in ROM-DOS.

ROM-DOS Developers Kit — $495 (30-Day Money Back Guarantee)

For free demo disk, info or orders, call **Datalight:**

1-800-221-6630

17505 - 68th Ave. NE, Bothell WA 98011
(206) 486-8086 FAX (206) 486-0253

CIRCLE NO. 265 ON READER SERVICE CARD

CIRCLE NO. 310 ON READER SERVICE CARD

of them equally. For example, the application may be heavily interrupt-driven, and the software may not use semaphores at all. The application's designer can compute the application-specific number if he/she knows (or can estimate) the relative frequency of different Rhealstone components in the application.

The steps for computing application-specific Rhealstones are as follows:

1. Measure the individual components (t1 to t6) as before.
2. Estimate the relative frequency of each Rhealstone component's occurrence when the application is executed, and assign nonnegative real coefficients (n1 to n6) proportional to the frequencies. For example, if interrupts occur five times more often than task switches do, and semaphores and intertask communication are not used in the application code, the value of n3 should be three times the value of n1, and n4 and n6 should be set to 0.
3. Compute a weighted average of the Rhealstone components:

```
t' = (n1*t1 + n2*t2 + n3*t3 + . . . . +
        n6*t6) / (n1 + n2 + n3 . . . + n6)
```

4. Invert the average to get the result 1/t' application-specific Rhealstones/second.

The procedure for computing generic and application-specific Rhealstones outlined in this article is different from that specified in the original proposal. The original procedure specified that each Rhealstone component should be arithmetically inverted separately and the average taken thereafter. I am grateful to the many readers who wrote to point out that, with the previous procedure, a computer system with high performance in one or two components and bad performance in the others would outshine one with moderately good performance in all categories.

The revised algorithm specifies that the components be averaged first and then arithmetically inverted (see acknowledgment 2). This algorithm ensures that if a real-time system shows bad performance in even one category, its overall score will suffer badly. This is intentional, because to guarantee quick response time, moderately good performance is needed in all Rhealstone categories. In other words, a real-time system that takes several seconds to respond to an interrupt will have a low Rhealstone rating, even if it delivers microsecond-range performance when switching context or exchanging semaphores.

Acknowledgment

1. Robert Wilson, GE Huntsville, Alabama, and Ketan Sampat, Intel Hillsboro, Oregon, for their input on measurement of inter-task communication performance.
2. Marco Pellegrino, Siemens AG Munich, Federal Republic of Germany, for suggestions on computation of Rhealstone performance number.

Availability

All source code is available on a single disk and online. To order the disk, send $14.95 (Calif. residents add sales tax) to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call 800-356-2002 (from inside Calif.) or 800-533-4372 (from outside Calif.). Please specify the issue number and format (MS-DOS, Macintosh, Kaypro). Source code is also available online through the *DDJ* Forum on CompuServe (type GO DDJ). The *DDJ* Listing Service (603-882-1599) supports 300/1200/2400 baud, 8-data bits, no parity, 1-stop bit. Press SPACEBAR when the system answers, type: listings (lowercase) at the log-in prompt.

DDJ

(Listings begin on page 100.)

Vote for your favorite feature/article.
Circle Reader Service **No. 4.**

PCYACC™
Version 2.0

PROFESSIONAL LANGUAGE DEVELOPMENT TOOLKIT

Includes "Drop In" Language Engines for SQL, dBASE, POSTSCRIPT, HYPERTALK, SMALLTALK-80, C++, C, PASCAL, PROLOG, FORTRAN, COBOL, and ADA.

PCYACC Version 2.0 is a complete Language Development Environment that generates ANSI C source code from input Language Description Grammars for building Assemblers, Compilers, Interpreters, Browsers, Page Description Languages, Language Translators, Syntax Directed Editors, Language Validators, Natural Language Processors, Expert System Shells, and Query languages.

Complete Grammars, Lexical Analyzers, and Symbol Table Management for ANSI C, K&R C, ISO Pascal, FORTRAN, dBASE III/Plus and IV, SQL, C++, Smalltalk-80, APPLE HyperTalk, C&M Prolog, YACC, LEX, FORTRAN-77, COBOL, ADA, and POSTSCRIPT are included. DOS, OS/2, QNX, SUN, AIX, and Macintosh versions are available.

Example application sources are provided to be used as skeletons for new programs. Examples include a desktop calculator, an Infix to Postfix Translator, a DBASE and SQL Syntax analyzer, an implementation of the PIC[ture] language, and a C++ to C translator.

NEW! Advanced ERROR HANDLING, RECOVERY, and REPORTING Provided

- Quick Syntax analysis option
- Optional Abstract Syntax Tree
- Manual "Compiler Construction with PC'S" included
- Lexical Analyzer generator ABRAXAS PCLEX is included
- Fully compatible with UNIX YACC grammars
- 30 day money back guarantee
- No charge for AIR Shipping anywhere in the world

Professional Version $495.00—Personal Version $249.00

ABRAXAS™ SOFTWARE, INC.
7033 SW Macadam Ave. Portland, OR 97219 USA
TEL (503) 244-5253 · FAX (503) 244-8375
AppleLink D2205 · MCI ABRAXAS

CIRCLE NO. 75 ON READER SERVICE CARD

Bounding Box Data Compression

An optimized font data compression method for fast screen I/O environments

Glenn Searfoss

The optimal data compression method for a given situation is determined by the type of data to be used and its intended application. To date, a variety of methods have been used to effect a balance between compression and display speed of bit-mapped font data.

One school of thought maintains that only noncompressed font data be used. While this data can be displayed rapidly, the amount of data storage required for fonts can be prohibitive, particularly when used with higher display resolutions and colors.

Another school stresses maximum compression of font data at all times. In this situation, each time a character is accessed, the cell data must be reconstituted. This saves data storage space but sacrifices access speed during screen display.

A third approach attempts to incorporate both aspects. Here font data remains compressed until accessed, at which point all data is uncompressed. A font may then be used at optimal speed. As additional fonts are uncompressed, the data storage limitation of the first method is encountered. This limitation can be ameliorated by each caching system in which only the most recently used characters are available in uncompressed form (at the expense of additional code complexity).

Because none of these methods are completely satisfactory for font screen display, it is important to develop a better balance between efficient access and efficient storage.

Glenn works at Data Transforms Inc., and can be reached at 616 Washington Street, Denver, CO 80203.

The terms "fast-access" or "on-the-fly" have been used to describe bit-mapped font data optimized for fast screen I/O. It is ultimately desirable in this environment to achieve maximum data compression while maintaining or increasing data access speed. To this end, it is essential to understand certain restrictions inherent with the graphics display of bit-mapped font data.

First, there must be minimal calculation of font data. Vector format and highly compressed bit-mapped fonts both require recalculation of character data at display time. This greatly restricts their usefulness in a fast-access environment. Speed optimization occurs when there is little or no data compression and reconstruction.

Second, character size must relate to the display resolution. As screen display resolution increases, larger characters are required to maintain the same relative size and quality as characters used on lower-resolution displays.

Third, font data storage requirements must not impinge upon code space. As bit-mapped fonts increase in size, their data storage needs quickly reach mammoth proportions. Realistically, some form of data compression is required for a program and several large font sets to coexist in memory.

Fourth, the amount of code necessary must be functional, effective, and preferably compact. Coding requirements are reduced with minimal data compression and decompression.

The data compression method best suited for this application will achieve a dynamic balance between these four criteria. The Bounding Box method of data compression is such a method.

To illustrate its effectiveness in this situation, a comparison with a commonly used method of data compression, run length bit encoding (RLE), is useful. This comparison is useful even though the two approaches are not mutually exclusive: One can store run length encoded characters in the Bounding Box format.

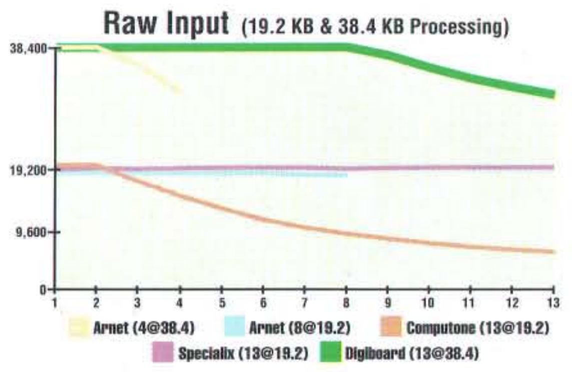
A valid comparison depends upon establishing a common reference point. Because noncompressed data is the basic requirement for both compression schemes, a brief outline of this font data format may be useful.

Noncompressed fonts are used as is. The font data is comprised of header information and complete character cell data. The header may detail as many character formatting aspects as desired.

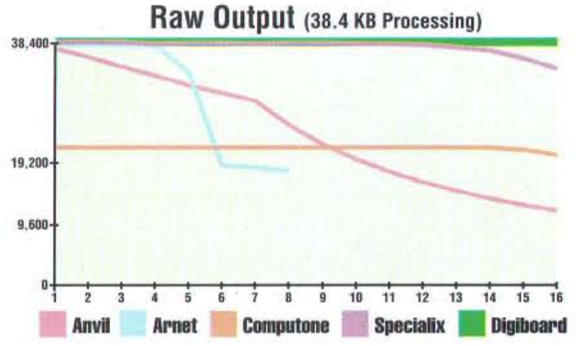
A typical header for standard noncompressed font data is shown in Listing One, page 108. In this case, I'm using the C language data structure of Data Transforms' Fontrix (Font1) Format.

Each font has a header that defines the font and points to the character bitmaps. Immediately following the font header is the character bitmap data. Character bitmaps are stored as scanrects in scanline order from top to bottom. Each scanline is stored byte wise left to right, left justified, and rounded to byte length. Each byte is stored 8 bits per byte where MSB is the leftmost pixel. Using this as a standard font header to describe Figure 1, we can proceed to specifics concerning methods of compressing this data.

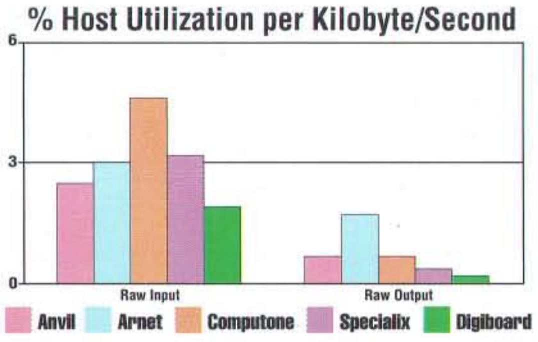
The new DigiCHANNEL series out-performs all other leading multi-user communications boards.



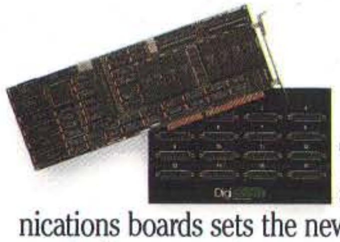
Raw Input: Primarily data received via host-to-host communications. The higher the better.



Raw Output: Processed data from host applications to terminal users (spreadsheet, word processing, etc.) The higher the better.



Processor Overhead: Percentage of host CPU time utilized for I/O processing tasks. The lower the better.



The new DigiCHANNEL series of multi-user communications boards sets the new performance standard for terminal response time, especially under heavy user-load conditions. The key to this performance is the synergy between our hardware and our new Front End Processing real-time Operating System (FEP O/S 5.4) software.

The proof is in the numbers, and a good example is the DigiCHANNEL PC/16i. In benchmark tests, it beats every other leading board in the two critical areas that determine board performance: *data throughput* and *processor overhead.*

Data throughput is calculated by measuring the total amount of data that a board can handle per port and per system. The higher the data throughput, the faster the response time for each user on the system.

Processor overhead is the amount of additional processing imposed on the CPU to handle the data input/output being controlled by the communications board. The less time the CPU needs to spend on I/O chores, the more time it can spend processing applications for terminal users.

Call for our FREE technical white paper with all the details on our benchmark testing. While you're at it, ask for our FREE booklet, *How to Do Multi-User Right.*

No matter how simple or complex your multi-user systems, you can trust DigiBoard to put you at the head of the pack. And keep you there.

DigiBoard
Plugging you into tomorrow

6751 Oxford Street • St. Louis Park, MN 55426
1-800-344-4273 • In Minnesota (612) 922-8055

CIRCLE NO. 376 ON READER SERVICE CARD

CODE RUNNER™

Create ultra compact and fast TSRs with full DOS access!

CodeRunneR provides the most powerful tools available for creating feature-packed, tiny TSRs with Turbo C® and Microsoft C®.

Now, on the heels of CodeRunneR's huge success, we are announcing our Professional Developers Kit series - PDK#1 and PDK#2.

CodeRunneR Features

- Function Level Granular
- Auto recovery of Initialization Code and Data
- Create MultiTasking programs
- Scheduler supports > 10,000 events
- Access TSRs via up to 256 hot-keys or interrupts
- BCD Floating Point Library
- Source for TSRs: Help, Programmers Calculator, Business Calculator, ASCII/Color, Date/Alarm & Trap/Debug

CodeRunneR $149.00

Call our BBS for FREE sample TSRs, benchmarks & support. See for yourself how powerful and tiny CodeRunneR TSRs can be. BBS: (508) 875-8009

PDK#1 Features

- Serial Communications Library, including background file transfers
- LIM Support for Code and Data
- Print SPOOLer functions
- Program Registration functions
- Free copy of HyperCOM (w/ source)

PDK#2 Features

- High speed floating point library
- 6x Borland and 2x ALTMATH

PDK#1&2 $99.00 each

CodeRunneR is great for non-TSR programming too. Excellent for creating MultiTasking programs!

NO ROYALTIES. MONEY BACK offer and source code available. Order today by Phone, FAX or BBS. MasterCard & VISA accepted.

Microsystems Software, Inc.™

600 Worcester Road
Framingham, MA 01701
Phone: (508) 626-8511
FAX: (508) 626-8515

CodeRunneR is a TM of Microsystems Software, Turbo C is a TM of Borland International and Microsoft is a TM of Microsoft Corporation

CIRCLE NO. 181 ON READER SERVICE CARD

(continued from page 56)

Bounding Box Compression

The Bounding Box Compression Method (see Figure 2) involves outlining a cell's "bit-on" character data with the smallest box possible. Coordinates that position the "box" relative to the original character cell are saved in the font header. This compression method is automatic within the Data Transforms Font Editor when a font over 32 × 32 pixels in size is saved. A sample header for a font compressed using the Bounding Box method is shown in Listing Two, page 108 (again using Data Transforms' Fontrix [Font2] Format).

The data listed in the *struct font2* portion of the (Font2) font structure above is defined as follows:

• The font header is the same as described in the *struct font1head* of the noncompressed font data format; font cell segments are an array of segment pointers to characters kept as offsets from the beginning of the font file. For example, if an array value for a character = 10, then the starting address of a character's "bounding box" data = (the start of file address + *size of [struct font2])* + (10 × 16), where 1 segment = 16 bytes.
• The horizontal size is the actual width in bits of the bounding box.
• The horizontal offset is the distance in bits from the left edge of the cell to the upper lefthand corner of the bounding box.
• Horizontal bytes refers to the actual size in bytes of the interior of the bounding box.
• The vertical size is the actual height in bits of the bounding box.
• The vertical offset is the distance in bits from the top edge of the cell to the upper edge of the bounding box.

The character data is never compressed; rather, the empty space outside the "bounding box" is discarded. The analogy of a shrinkwrap bag can be used to illustrate.

Imagine a character cell placed within a shrink-to-fit bag. The noncompressed cell is the unshrunk bag. Now shrink the bag until all edges contact the outer limits of bit-on data, forming a rectangular bounding box. For some characters — such as a lowercase "i" — this correlates to a major amount of shrinkage, and the shrinkage correlates to saved data space, hence, compression. An uppercase "W" may fill most of a cell. In this instance minimal shrinkage will occur, with little or no saved data space. However, the overall net savings in data space for an entire font set will be great because few characters fill an entire cell (see Figures 2, 3, and 4).

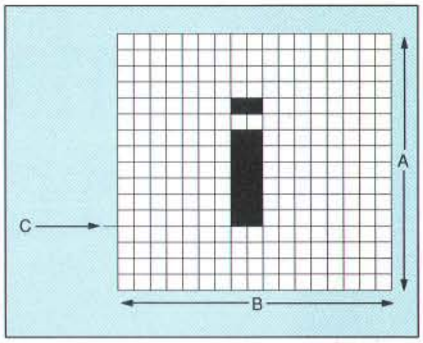


Figure 1: *A noncompressed character cell.* ***A.*** *Vertical cell size 16 pixels.* ***B.*** *Horizontal cell size 17 pixels.* ***C.*** *Baseline character cell 12 pixels*

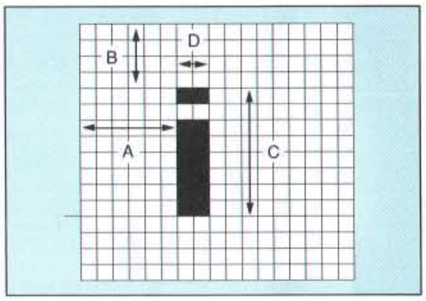


Figure 2: *A character cell compressed using the Bounding Box method. The data listed is saved for each character when the font was created and stored in a look-up table in the font header.* ***A.*** *Horizontal offset 6 pixels.* ***B.*** *Vertical offset 4 pixels.* ***C.*** *Vertical size of bounding box 8 pixels.* ***D.*** *Horizontal size of bounding box 2 pixels*

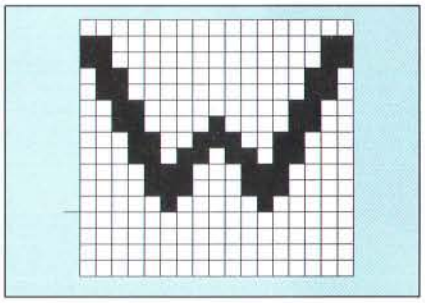


Figure 3: *Noncompressed character cell. Vertical cell size 16 pixels. Horizontal cell size 17 pixels. Baseline character cell 12 pixels*

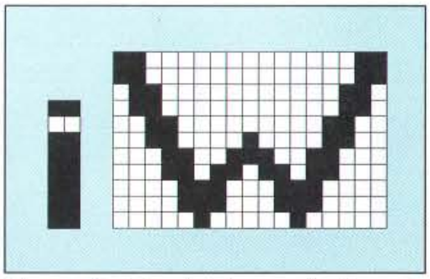
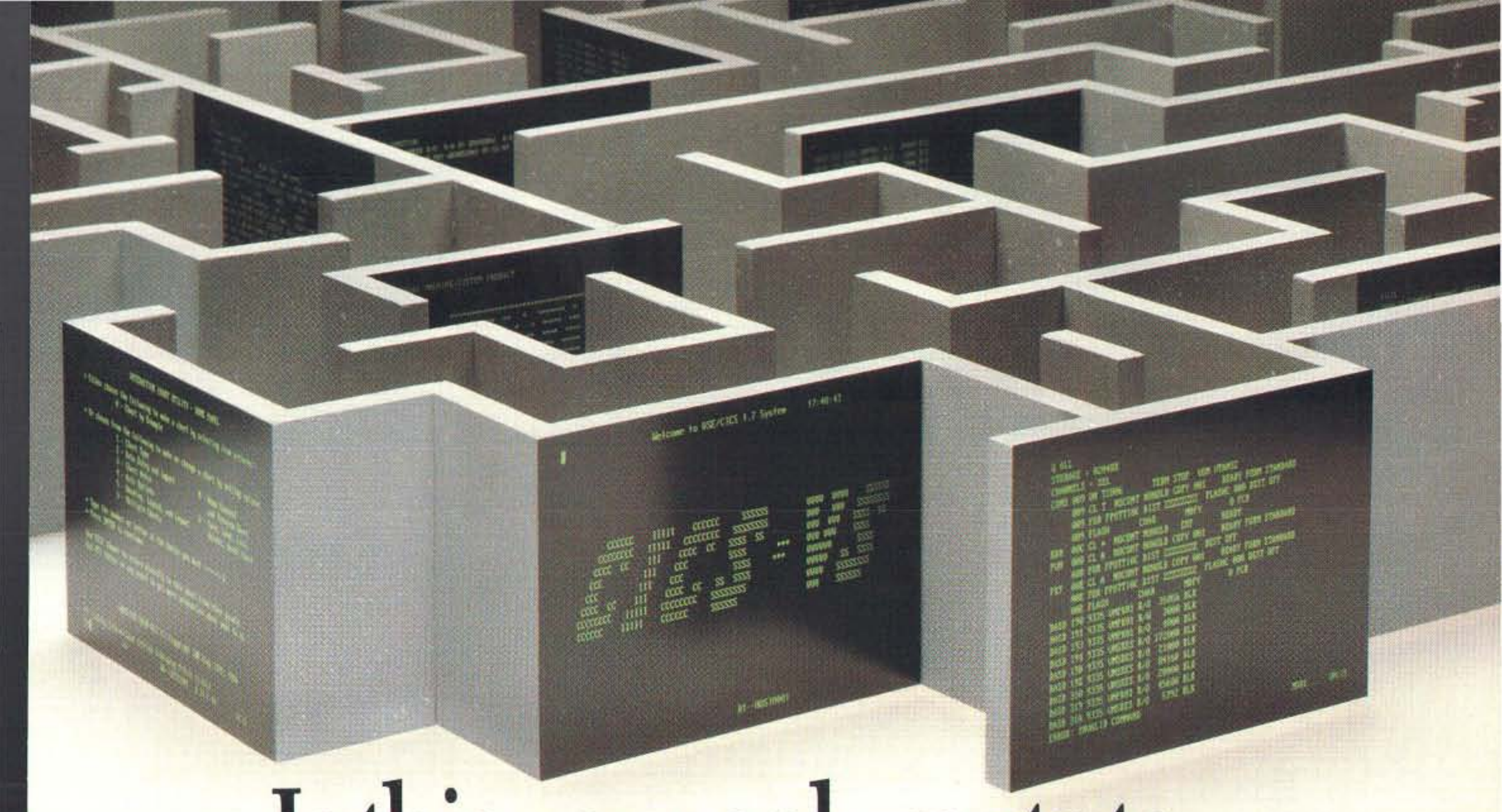
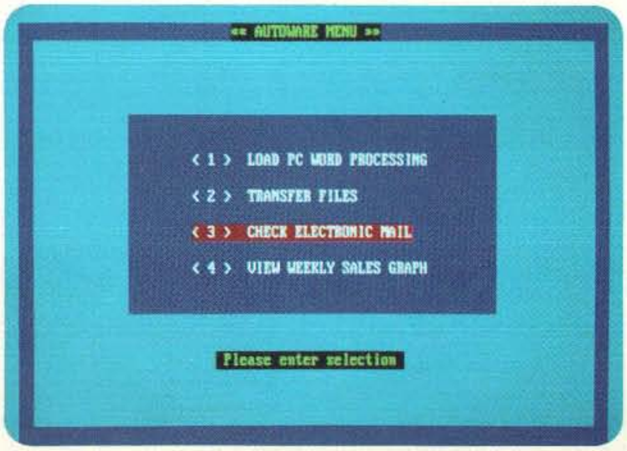


Figure 4: *Actual data saved when a character cell is compressed using the Bounding Box method*



Is this your only route to mainframe information?



<< AUTOWARE MENU >>
< 1 > LOAD PC WORD PROCESSING
< 2 > TRANSFER FILES
< 3 > CHECK ELECTRONIC MAIL
< 4 > VIEW WEEKLY SALES GRAPH
Please enter selection

Escape to Autoware: NOW!

When you need mainframe access, why endure a frustrating labyrinth of screens? Especially when Attachmate software delivers simple single-menu access.

It frees you to select E-mail, transfer files and retrieve data with single-keystroke ease. It's so automated, we call it Autoware. So fast, we had to call it NOW!

NOW! lets you customize menus for specific procedures or applications, such as unattended file transfer. Within minutes, even non-programmers can automate most repetitive mainframe chores.

Make mainframe access a direct path, not a mindless maze. Let Autoware do the work *for* you—NOW! Call for your free demo disk: **800-426-6283.**

Attachmate

Attachmate Corporation 13231 S.E. 36th Street Bellevue, WA 98006 (206) 644-4010

NOW! and Autoware are trademarks of Attachmate Corporation

CIRCLE NO. 188 ON READER SERVICE CARD

(continued from page 58)

Run Length Bit Encoding

A RLE font would possess a header similar to the font header described in the "struct font1head" of the noncompressed font data format. The main differences between RLE and the Bounding Box lies in the method of character data storage and how code points to it.

In a simple case of run length bit encoding, bit mapped data is compressed by reading each scanline of a character cell, grouping adjacent bit-on or bit-off data and saving the information as pairs of ASCII digits. For example, by using RLE the marked scanline in Figure 5 could be compressed as follows:

- 0×06 (6 identical bits in a row), 0×00 (those bits are bit-off data)
- 0×02 (2 identical bits in a row), 0×01 (those bits are bit-on data)
- 0×09 (9 identical bits in a row), 0×00 (those bits are bit-off data)

Having established the basic structure of the two methods, it now remains to compare their differences in access speed, compression efficiency, and coding requirements.

Speed of Compressed Data Access

When a Bounding Box compressed character is used, there is no run-time penalty during screen display. A character cell is not reconstituted. Rather, the character data is used as is and positioned relative to the original cell size information.

Fonts compressed with run length bit encoding pay a run-time penalty during screen display. If a font remains compressed while being accessed, each time a character is displayed to the screen, the entire character cell must be reconstructed. In a case like this, you'll be watching the clock.

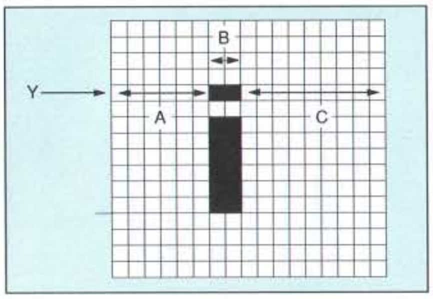


Figure 5: *Run length bit encoding for scan line Y. Each portion of data is compressed into a pair of ASCII digits.* ***A.*** *0x06 0x00* ***B.*** *0x02 0x01* ***C.*** *0x09 0x00*

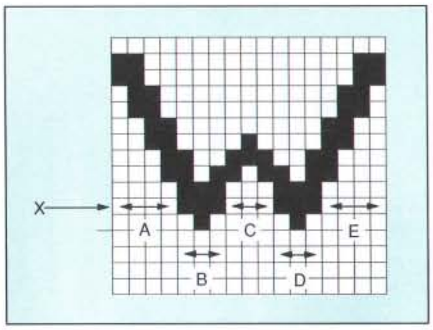


Figure 6: *Run length bit encoding for scan line X. Each portion of data is compressed into a pair of ASCII digits. The more data dispersal on a scan line, the more detailed this method of compression becomes.* ***A.*** *0x04 0x00* ***B.*** *0x03 0x01* ***C.*** *0x03 0x00* ***D.*** *0x03 0x01 E. 0x04 0x00*

Compression Efficiency

The Bounding Box routine can run at up to 90 percent of the overall compression efficiency of more computationally expensive compression algorithms. The actual character bits-on data is never compressed.

For this reason, the Bounding Box approach is more properly called a technique or a character storage format, rather than an algorithm. Information regarding the bounding box (size, [x,y] position within the original cell, and so on) is kept for each character in a lookup table in the font header (see Figures 2 and 4).

Using this compression method on the character cell in Figure 1 (lowercase i), the net gain in savings is 94 percent of the original character cell size. For Figure 3 (uppercase W), the net savings is 31 percent of the original character cell size. The percentage in savings correlates to the discarded zero (bit off) data outside the bounding box (see Figure 4).

The run length bit encoding (simple case) compression method can run at up to 98 percent of overall compression efficiency, but will be computationally expensive. Using this compression method on the character cell in Figure 1 (lowercase i), the net gain in savings is 78 percent of the original character cell size. For Figure 3 (uppercase W), the net savings is 57 percent of the original character cell size (see Figures 5 and 6). Table 1 provides a complete RLE table for Figure 5. Read

World Class Software Security

The parallel port interface (PPI) connects between the printer port on a PC and the printer cable. The PPI holds two Key Tags, one on each side. Each Key Tag contains a secure custom chip which is pre-programmed by Glenco to only work with the assigned software package. A second Key Tag can be employed to protect another package, or may be used to turn other software packages "on", remotely or on-site.

- STANDARD KEY TAG - Software is protected for an unlimited number of executions. They are pre-programmed to include a sequentially assigned S/N.
- COUPON KEY TAG - Software is valid for a preset number of executions. The Coupon count can be reset remotely or on the customers site by using a second update Key Tag.
- READ/WRITE KEY TAG - With programmable memory. Perfect for companies which have multiple products or a product with several optional modules. By having several packages protected using one Key Tag, your costs are lowered.
- DURATION KEY TAG - Has a clock on board. (Available late '89)

Secure software and data with reliable, effective protection products that won't burden honest users.

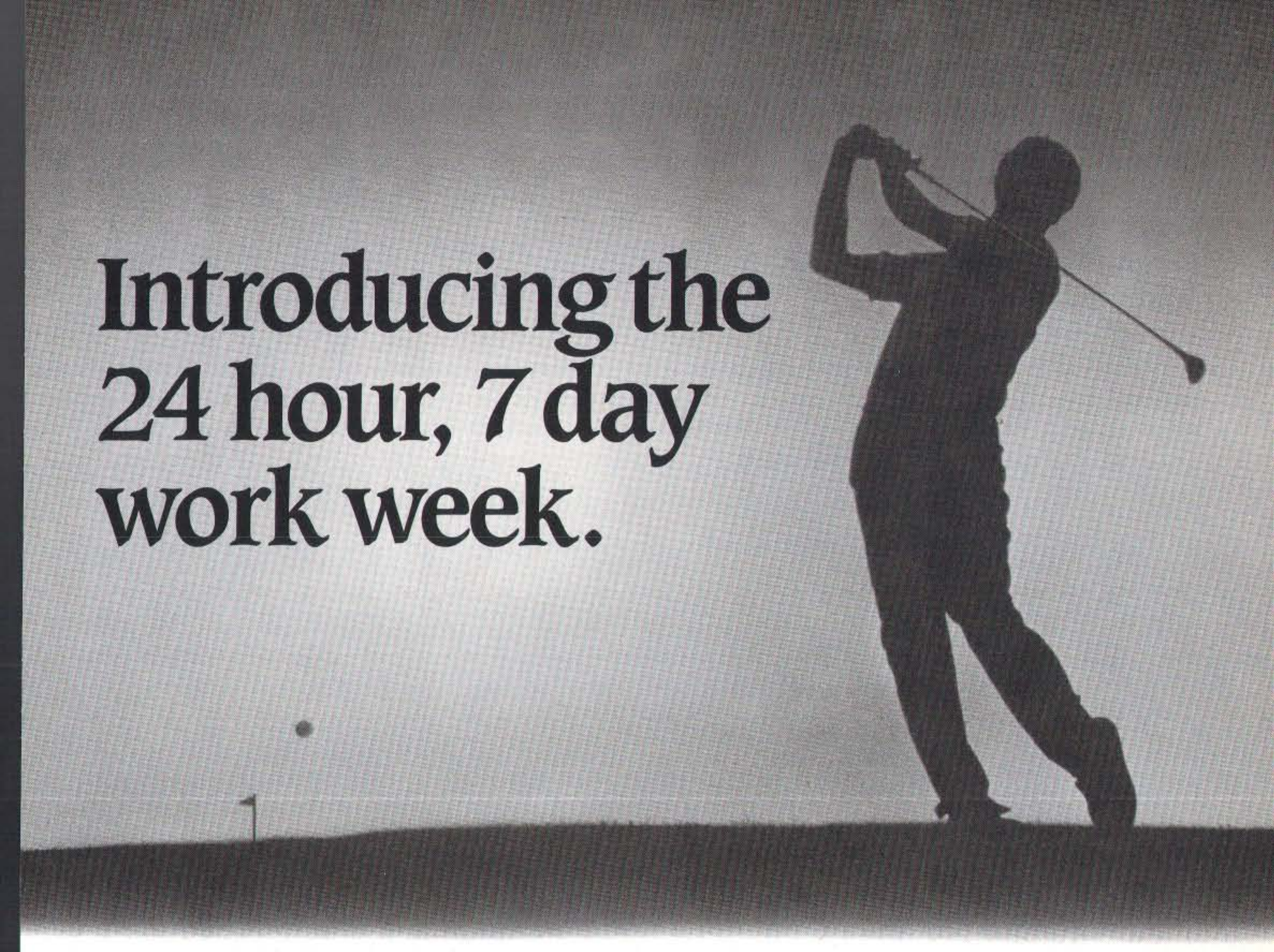
Glenco is a world leader in the area of software security products and services. Our copy protection products and data security products are second to none. They are designed to function on a wide variety of third party hardware. We have over 3500 satisfied software firms utilizing our products. We also have a full line of disk based protection systems.

- MACHINES SUPPORTED - IBM PC/XT/AT & PS/2, Macintosh
- OPERATING SYSTEMS - MS-DOS, XENIX, Network, Finder, & Multifinder.
- LANGUAGES/COMPILER - Over 50, including runtime packages, data bases and spread sheets. We have a non-programmers interface as well.

Call or write for more information.

GLENCO ENGINEERING INC.
SERVING THE SOFTWARE INDUSTRY SINCE 1979
721 W. Algonquin Road, Arlington Hts., IL 60005, (312) 364-7638, FAX 364-7698
In Europe contact: SDC Security Systems, The Netherlands
Tel: +31-45-441535, FAX: +31-45-444747

CIRCLE NO. 225 ON READER SERVICE CARD



Introducing the 24 hour, 7 day work week.

Automator, the software "robot" that never sleeps.

In today's competitive global economy, optimizing the productivity of your corporate-wide information system has become critical.

The flow of information between your mainframe and workstations should be serving you at full capacity, around-the-clock, freeing managers for more creative pursuits.

Automator was developed to simplify the process of linking PC workstations into any information network. It is a complete software system for creating and running unattended operations which drive the PC workstation, using pre-programmed instructions as if the operator was actually there.

Automator can faithfully perform all those day-to-day tasks that must be done—like downloading files for distribution on E-mail; automating month-end procedures; searching and capturing mainframe data for use in a PC application; constant monitoring of the computer network—and 101 other uses.

Universal, Productive, Efficient

Automator software makes emulation specific APIs redundant, because it can work concurrently with most DOS applications, LANs and terminal emulations, including 3270, 5250 and VT 220.

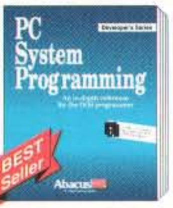
The software's Program Generator rapidly produces applications, by simply watching a PC's screen, keyboard and clock. And with its high level programming language, can give developers the flexibility to build sophisticated applications that can be tested in a matter of hours.

And even though *Automator* has the capacity to perform multiple operations, its unique technology keeps DOS memory requirements down to a minimum of only 32K.

For more information about *Automator*, call Mark Gillett or Harold Lund at **212-475-2747** or **800-992-9979.**

AUTOMATOR
From Direct Technology 10 East 21st Street, New York, NY 10010

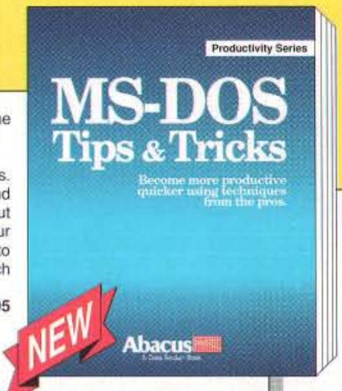
We Sell Know-How



PC System Programming for Developers is an encyclopedia of PC technical and progamming knowledge. Features parallel working examples written in Pascal, C, assembly and BASIC. Explains memory layout, DOS operations, using extended, expanded memory, writing device drivers, hard disks, PC ports, mouse drivers, fundamentals of BIOS, graphics and sound, TSR programs, complete appendices. 920+ pp w/2 disks, over 1 meg of programs in compressed format.
ISBN 1-55755-036-0. 920+ pages with 2 disks. **$59.95**

MS-DOS Tips & Tricks shows you how to become more productive quickly with DOS.

Contains dozens of hints and tips from DOS experts. Learn how to find any file on hard disk quickly and painlessly; how to copy data from a backup without the RESTORE command; learn to protect your data and PC from unauthorized access; how to cold-start your PC from a batch file and much more.
ISBN 1-55755-078-6. 210 pages. **$17.95**
Companion disk available: $14.95



PC File Formats and Conversions is for every PC user who ever wanted to know how to exchange file formats. Describes in detail file formats for major software applications. Use IMPORT/EXPORT functions in many applications. Includes conversion program. Companion disk included.
ISBN 1-55755-059-X. 280 pages. **$34.95**

PC TOOLS Complete - The total reference guide to the best-selling PC TOOLS Deluxe software. Shows you how to master all of the utilities and applications, many techniques, hints and tips. Learn advanced features that will make your computing sesions easier and faster. Covers Version 5.5 of PC TOOLS Deluxe.
ISBN 1-55755-076-X. 450 pages. **$22.95**

Tips & Tricks for your PC Printer is subtitled "understanding and using your PC printer more effectively". Takes the hassle out of working with PC printers. You'll learn how to setup and connect your printer. You'll learn to use the many features that are built into the popular printers, but rarely used because they're so difficult to understand. The companion disk has several practical printer utilities: online printer HELP, printer control aid, printer font editor and more.
400 page book includes companion disk.
ISBN 1-55755-075-1. Suggested retail price: **$34.95**

PC TOOLS Companion Quick Reference - A durable, hardcover guide filled with all of the essentials for using PC TOOLS Deluxe Version 5.5 more effectively. Contains the correct syntax organized logically for you to quickly and easily get your job done.
ISBN 1-55755-012-3. Hardcover. 220 pages. **$12.95**

Coming in April Turbo Pascal Internals for Developers.
Available at B Dalton Booksellers, Waldens, and Software Etc. nationwide. In the UK contact Computer Bookshops 021-706-1188. In Canada contact Addison Wesley 416-447-5101.

In US and Canada call TOLL FREE: 1-800-451-4319

Abacus
Dept. D4, 5370 52nd Street SE • Grand Rapids, MI 49512
Phone: (616) 698-0330 • Fax: (616) 698-0325
In US and Canada add $4.00 postage & handling. Foreign orders add $12.00 postage per book. We accept Visa, Master Card or American Express. Call or write for your free catalog of more PC books.

CIRCLE NO. 82 ON READER SERVICE CARD

(continued from page 60)
each scan line left to right, top to bottom, numbered 0 to 15. Table 2 is the complete RLE table for Figure 6. Again, read each scan line left to right, top to bottom, 0 to 15.

Coding Requirements

The Bounding Box method is invoked at a time when display speed is not an issue — during font creation. All the data needed to use a character is saved in a lookup table in the font header at this time. Because the actual data within the bounding box is not compressed, no coding is required to reconstruct a character cell. Code need only index into the data and index the screen pixel position.

Run length bit encoding requires code to do more than access font data. It must also handle the peculiarities involved with compressing and uncompressing data. Font characters that remain compressed while being used must be reconstructed each time they are accessed. Font data that is reconstituted once and thereafter accessed as noncompressed data, can instigate a data storage conflict. As additional fonts are uncompressed, their combined data storage requirements begin to compete with code and operational program space.

The Nitty Gritty

The Bounding Box method is most effective when the data to be compressed is bit-mapped data; all bit-on data is concentrated and confined in one area, as is often the case with alphanumeric character fonts; the ratio of bit-off to bit-on data is high; fast usage of compressed data is of most importance (for example, screen I/O, graphics printing, and so on); and it is preferred that font data remain compressed when being accessed.

```
 0] 0×17 0×00
 1] 0×17 0×00
 2] 0×17 0×00
 3] 0×17 0×00
 4] 0×06 0×00 0×02 0×01 0×09 0×00
 5] 0×17 0×00
 6] 0×06 0×00 0×02 0×01 0×09 0×00
 7] 0×06 0×00 0×02 0×01 0×09 0×00
 8] 0×06 0×00 0×02 0×01 0×09 0×00
 9] 0×06 0×00 0×02 0×01 0×09 0×00
10] 0×06 0×00 0×02 0×01 0×09 0×00
11] 0×17 0×00
12] 0×17 0×00
13] 0×17 0×00
14] 0×17 0×00
15] 0×17 0×00
```

Table 1: *RLE table for Figure 5. Reading each scan line left to right, top to bottom, and numbered 0– 15*

```
 0] 0x17 0x00
 1] 0x02 0x01 0x13 0x00 0x02 0x01
 2] 0x02 0x01 0x13 0x00 0x02 0x01
 3] 0x01 0x00 0x02 0x01 0x13 0x00 0x02 0x01 0x01 0x00
 4] 0x01 0x00 0x02 0x01 0x13 0x00 0x02 0x01 0x01 0x00
 5] 0x02 0x00 0x02 0x01 0x13 0x00 0x02 0x01 0x02 0x00
 6] 0x02 0x00 0x02 0x01 0x04 0x00 0x01 0x01 0x04 0x00 0x02 0x01 0x02 0x00
 7] 0x03 0x00 0x02 0x01 0x02 0x00 0x03 0x01 0x02 0x00 0x02 0x01 0x03 0x00
 8] 0x03 0x00 0x02 0x01 0x01 0x00 0x02 0x01 0x01 0x00 0x02 0x01 0x01 0x00 0x02 0x01 0x03 0x00
 9] 0x04 0x00 0x03 0x01 0x03 0x00 0x03 0x01 0x03 0x00
10] 0x04 0x00 0x03 0x01 0x03 0x00 0c03 0x01 0x04 0x00
11] 0x05 0x00 0x01 0x01 0x05 0x00 0x01 0x01 0x05 0x00
12] 0x17 0x00
13] 0x17 0x00
14] 0x17 0x00
15] 0x17 0x00
```

Table 2: *RLE table for Figure 6. Reading each scan line left to right, top to bottom, and numbered 0– 15*

GIVE YOUR PROGRAMS SOME ROOM FOR IMPROVEMENT

Developers today need all the room they can get. But within the confines of 640K, space can be hard to come by. Even something as essential as your make utility can become a hindrance due to its large memory requirements. If you're tired of running into the 640K wall, you need **Dr. Switch**™–the incredible memory management utility that allows you to push stand-alone and RAM resident programs out of the way while you compile, link, and test even the largest programs.

Let nothing stand in your way.
With Dr. Switch you can swap the RAM resident programs you rely on, to and from expanded memory, extended memory or hard disk so that you can get on with the business at hand. The Doctor works lightning-fast on everything from on-line help systems like the Norton Guides® and Peabody® to desktop utilities such as Sidekick®, Lotus Metro®, and PC Tools®–not to mention terminal emulators like DCA's e78 and IBM's PC Support program.

Make less of Make.
Dr. Switch reduces the overhead of Microsoft Make and Polytron's PolyMake from over 100K to a mere 4K, giving you the memory you need to run your compiler and linker more efficiently. So go ahead, swap your Make utility out of memory while you compile and link–Dr. Switch swaps it back, in a flash, when you're done!

Turn your editor into an integrated environment.
You read right. With Dr. Switch, you'll be able to compile and link directly from your editor. And you can feel secure about it too! If you run your program and it crashes, the Doctor will return you right to the program that you started from.

You'll hardly know it's there.
Unlike other programs that claim to give you extra memory, but actually take up large blocks of it themselves, Dr. Switch always keeps a low profile. It requires less than 4K and can take full advantage of any expanded or extended memory you have available. It's completely transparent, requires no setup, and is network aware. Like any good doctor, it can be called on at anytime from anywhere.

Don't just take our word for it.
Dr. Switch is the people's choice:
"Can't break the 640k barrier, work around it with Dr. Switch."
PC Magazine, March 28, 1989
Winner of the 1989 Data Based Advisor Readers Choice Award for Best Database Utility.

Attention Dbase developers
Our special Dbase DeveloperPak includes all the required royalty-free runtime modules, so that you can include the Dr. Switch technology along with the applications you distribute.

With the Doctor on call, you're in control.
When you've got RAM resident programs to swap or large applications to run, remember–there's nothing like having a doctor in the house.

The Doctor is in.

CALL AND PLACE YOUR ORDER TODAY!
212-787-6633

DR. SWITCH
$59.95*

DBASE DEVELOPERPAK
$99.95*†

Black & White International Inc.
P.O. Box 1040
Planetarium Station
New York, NY 10024-1040

Dr. Switch is a trademark of Black & White International, Inc. All others are trademarks or registered trademarks of their respective holders.

*Plus Shipping/Handling: U.S. orders add $6.00. Canadian and Foreign orders add $15.00. COD add $3.50. NY residents add sales tax. All payments U.S. funds/U.S. Banks only!

†Includes royalty-free runtime version.

CIRCLE NO. 407 ON READER SERVICE CARD

(continued from page 62)

Run length bit encoding is most effective when all bit-on data is widely dispersed or present at a majority of the cell edges; the ratio of bit-off to bit-on data is low; and the usage of data storage must be maximized or is of higher priority.

RLE and the Bounding Box are two good methods of data compression. Each has limits in its ability to handle graphics information. The decision to use a RLE or Bounding Box compression scheme should be based on the likely distribution of bit-on data and the preferred ratio of data compression to access speed.

As mentioned earlier, in situations where memory is extremely tight, the Bounding Box method can profitably be used in conjunction with RLE or other algorithms. Of the various compression methods currently used in fast-access screen I/O environments, the Bounding Box method is recommended for maintaining maximum data compression while allowing the greatest access speed of bit-mapped font data.

Availability

All source code is available on a single disk and online. To order the disk, send $14.95 (Calif. residents add sales tax) to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call 800-356-2002 (from inside Calif.) or 800-533-4372 (from outside Calif.). Please specify the issue number and format (MS-DOS, Macintosh, Kaypro). Source code is also available online through the *DDJ* Forum on CompuServe (type GO DDJ). The *DDJ* Listing Service (603-882-1599) supports 300/1200/2400 baud, 8-data bits, no parity, 1-stop bit. Press SPACEBAR when the system answers, type: listings (lowercase) at the log-in prompt.

DDJ

(Listings begin on page 108.)

Vote for your favorite feature/article.
Circle Reader Service **No. 5.**

CODECHECK CODECHECK CODECHECK

CODECHECK® Version 2.0

PROFESSIONAL SOURCE CODE ANALYST

Includes "Drop-In" Rules for compliance analysis, Adherence to specifications, Measures of complexity, Silent error detection, Code maintainability, and Portability.

CODECHECK Version 2.0 is a *forward chaining expert system* with rule compilation for fast execution speed. Rules are written in a C subset, including *if—*, *for—*, and *while—* constructs. The system can generate prototypes from old-style function definitions and can detect many silent errors not found by today's compilers. Additionally, CODECHECK will verify that identifiers comply with user defined and type encoded spelling rules and will perform full "Lint" error detection with user specified messages. CODECHECK measures code size and complexity with standard or user designed metrics and detects violations of standards for C coding style.

- **Maintainability-CODECHECK identifies complex, sloppy, and hard to maintain code.**
- **Portability-CODECHECK identifies code that will not port between DOS, OS/2, Unix, and the Macintosh, and checks for compatibility with ANSI and other standards.**
- **Manual "C CODE ANALYSIS USING CODECHECK," is included.**
- **No charge for Shipping anywhere in the world.**
- **30 day money back guarantee.**

CODECHECK is a complete Source Code Analysis Environment designed to solve your software engineering problems. Expert Rule Files are included for: 1) Compliance with generic corporate C coding standards, 2) Adherence to specifications for type encoded spelling names, 3) Measures of complexity (McCabe Cyclomatic, Halstead and code density), 4) Advanced silent error detection (macro definition and expansion, variable argument functions, namespace ambiguities, *printf* and *scanf* formats), 5) Code maintainability (detects noncompliance with generic stylistic standards, overly complex expressions and uncommented functions and declarations; enforces strict prototyping and monitors compliance with size standards for statements, functions and modules), and 6) Portability (to Kernighan & Ritchie and Harbison & Steele compilers, ANSI C, Microsoft C 6.0, Quick C 2.0, Turbo C, VAX C, AT&T C++ and Apple MPW C).

To order, please call:

1-800-347-5214

DOS $495. OS/2 $695. Macintosh $495. UNIX $995.

ABRAXAS™ Software, Inc.

7033 SW Macadam Ave., Portland, Oregon 97219 USA
AppleLink D2205 · MCI ABRAXAS
TEL (503) 244-5253 · FAX (503) 244-8375

CIRCLE NO. 58 ON READER SERVICE CARD

WHY YOU WANT BATCOM!

This is definitely a fun product. John Dvorak, PC Magazine Nov. 28, 1989.

Wenham Software's *Batcom* is a batch file compiler that transforms your normal DOS batch files into ".exe" files. Compiled programs execute much **faster** than normal batch files, **protect your source code**, **add additional capabilities** to DOS, and give your batch programs a touch of professionalism never before possible!

Batcom compiles your DOS batch files with a **simple one step command**. Just typing "batcom demo" will compile your batch file named *demo.bat* into a file named *demo.exe*. That's it! Few or no changes to your existing batch file programs are necessary.

Speed: *Batcom* speeds your DOS batch files by interpretting each command before it needs to be executed. DOS must read each statement from disk as it is executed; compiled batch programs load once and go. This eliminates the disk thrashing that is common to many batch file programs. Each command is then executed in an efficient computer readable code rather than interpretted. The result is batch file programs that execute up to **4 times faster** (and more) than normal batch files.

Profit: Compiled batch files are your property, and you may distribute them without restrictions and without paying royalties.

apabilities: *Batcom* **extends DOS with over 60 new commands**:

- Use your own user defined variables in addition to DOS command line parameters and environment variables.
- Perform arithmetic.
- Read and test for keyboard input.
- Examine how much Lotus/Intel/Microsoft expanded (LIM EMS) memory your computer has.
- Examine the system date and time.
- Examine the current directory name.
- Position the cursor.
- Make TSR programs! (with optional add-on — $25).
- 'While' looping.
- Execute subroutines.
- Easily use ANSI control sequences.
- Perform powerful string operations.
- Read information from the screen.
- **AND MUCH MORE!**

Size: *Batcom* compiles your batch files to small programs. A menu program that gives the user a choice of running one of four programs compiles to under 4000 bytes of memory. Try that with any other programming language!

If you need to squeeze more performance from your existing batch files, or you no longer use DOS batch files because they aren't powerful enough, you need Wenham Software's *Batcom*.

Dealer inquiries invited.

YES! Send me Batcom NOW!

Name: ____________ Company: ____________
Address: ____________
City: ____________ State: ____________ Zip: ____________

Send me ____________ at $59.95 each. Total: ____________

☐ Check or money order enclosed ☐ Visa ☐ MasterCard
Account # ____________ Exp. Date ______ – ______
Name of Card Holder: ____________

Send to:

↗→↘ *Wenham Software Co.*
↑ • ↓ *5 Burley St.*
↖←↙ *Wenham, Ma. 01984*

Or call: (508)–774–7036

CIRCLE NO. 416 ON READER SERVICE CARD

1989 DDJ INDEX

C SOURCE

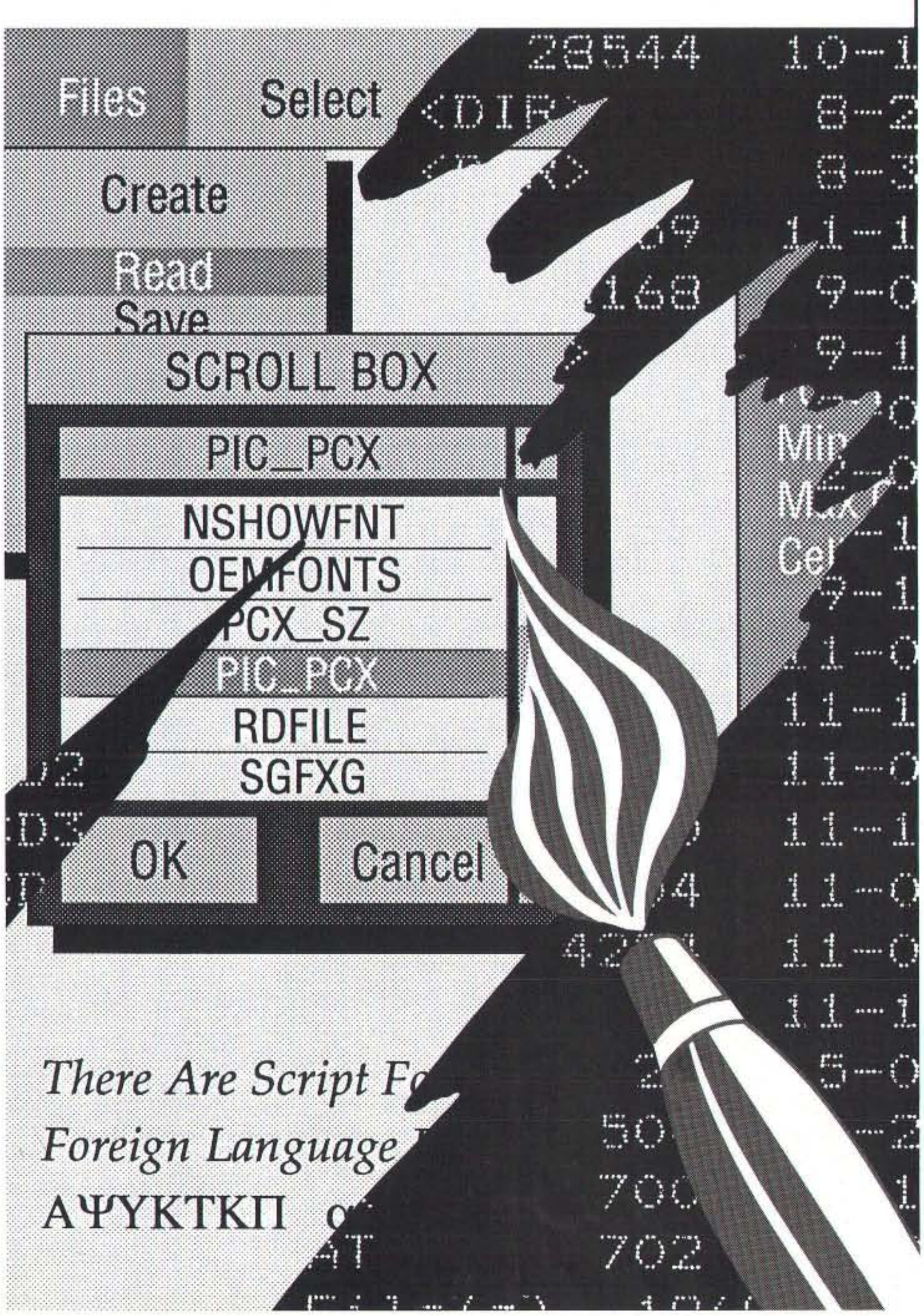
The GFX Libraries

NOW IN C, PASCAL & MODULA 2

The three GFX libraries have been translated from C to Pascal and Modula 2. No memory-hog TSRs to eat precious ram. Just small, fast linkable libraries. Each library stands alone. Use them together or with other graphics libraries. And of course, we provide all the source code.

Common features include:

Source Code: in your language.
No Royalties: for straight application programs.
Video Modes: Hercules, CGA, EGA, VGA (up to 800x16x16).
Supported Compilers: C — Microsoft, Borland, Lattice, Metaware, Zortech; **Pascal** — Borland (5.5); **Modula 2** — JPI and Stony Brook.



GFX Fonts & Menus Library

Now you can add a slick graphics user-interface to your programs: pull-down menus driven by mouse or hot-keys; dialog boxes; forms boxes; context-sensitive help; huge font selection (100+ fonts).
List Price: Source code - $150

GFX Font & Icon Editor

Not a library, but a fully blown editor. Make new fonts or modify fonts provided with GFX Fonts & Menus Library. Create new fonts or bitmapped icons. Use all 255 characters for fixed or proportional width fonts. Source code is only in C.
List Price: Source code - $100
Executable - $75

GFX Graphics Library

More power and speed than your compiler's library. Use virtual colors and auto-scaling to write code that adapts to any video mode at run-time. You also get rubber-banding and multiple viewports with automatic clipping and interger and floating-point scaling.
List Price: Source code - $150

GFX Screen Dump Library

Use this library to dump a graphics screen to a printer. We support most dot matrix, laser and color printers. Add new printers by filling in a table. Stretch the output to adjust for differences between video and printer resolutions.
List Price: Source code - $100
Executable - $75

FREE BBS

Extensive demos & info on all our products from a 24hr BBS.

BBS: (816) 478-0944
8 Data/1 Stop/No Parity

SPECIAL OFFER

Call us for special pricing for ordering direct from C Source.

Get our brochure on these and other programmers products.

**C Source, Inc.
400 N.E. Point Dr.
Lee's Summit, MO 64064
TEL: (816) 478-1888
FAX: (816) 478-3133**

CIRCLE NO. 350 ON READER SERVICE CARD

1989 DDJ INDEX

Compare Lattice Performance, Features, & Value!

LATTICE C DEVELOPMENT SYSTEM FOR DOS & OS/2

- Runs under DOS or OS/2 to create DOS, OS/2, and Family Mode Programs
- C language extensions for added versatility
- Global Optimizer allows your program to run up to 30% faster
- OS/2 API Library and Bind Utility eliminate the need for OS/2 developers kit

ONLY $250.00

LATTICE 80286 C DEVELOPMENT SYSTEM FOR DOS & OS/2

- Runs under DOS or OS/2 to create DOS, OS/2, or Extended DOS Programs
- 80286 DOS Extender with ***no run-time royalties!***
- Creates programs of up to nearly 16 MB
- Extended DOS Configuration Optimizer assures maximum performance
- Compiler, Debugger, Libraries, and Tools Performance Tuned for 286 Systems
- Improved debugging environment speeds program development

ONLY $495.00

LATTICE 80386 C DEVELOPMENT SYSTEM FOR DOS & OS/2

- Runs under DOS or OS/2 to create OS/2 or Extended DOS Programs
- Creates Programs using full 32-bit power
- 80386 DOS Extender with ***no run-time royalties!***
- Compiler, Debugger, Libraries, and Tools Performance Tuned for 386 Systems
- Compatible with PharLap DOS Extender

ONLY $895.00

CALL OR WRITE TODAY FOR A FREE CATALOG LISTING ALL LATTICE TOOLS.

CIRCLE NO. 140 ON READER SERVICE CARD

Lattice Enriches Your Environment

Lattice offers programming tools that let you take full advantage of today's high-powered hardware and operating environments. Each Lattice Development System contains the benchmark-winning Lattice Optimizing C Compiler, CodePRobe source-level debugger, LASM macro assembler, hundreds of Library Functions, and dozens of Powerful Tools — all specifically tuned for your programming environment.

When your programming environment changes, Lattice is there with on-going support. With 100% ANSI compatibility, programs written in Lattice C are highly portable. And only Lattice offers such a large selection of C compilers, C cross compilers, and embedded systems C compilers for a wide range of host/target hardware and operating systems.

Order a Lattice C Development System Today! Each Lattice product comes with a 30-day, money-back satisfaction guarantee plus unlimited free technical support. Lattice products are available from dealers and distributors worldwide or directly from Lattice. Call 1-800-444-4309 Today!

Lattice

Professional Programming Tools Since 1981
Lattice, Inc.
2500 S. Highland Avenue Lombard, IL 60148
Telephone: 708/916-1600 Facsimile: 708/916-1190

The joy of C-scape

The C-scape™ Interface Management System frees C programmers from the tedium of coding windows, menus, data validation, help, and text editing functions.

Moreover, C-scape is a joy to use. With C-scape's object-oriented design, you'll build more functional, more flexible, more portable, and more unique applications—and you'll have more fun doing it.

The industry standout. Many thousands of programmers have quit home-grown libraries and cumbersome, inflexible products for the pleasure of C-scape. The press agrees: "C-scape is by far the best . . . a joy to use," wrote IEEE *Computer*. *PC Magazine* chose C-scape to produce its Laboratory Benchmark Series 5.0 software because C-scape offers mouse support. Moreover, C-scape simultaneously combines text and graphics. And because C-scape makes it easy to create your own custom routines, major companies have selected C-scape as a standard for software development.

C-scape is built around an open architecture, so you can use it with data base management or other C libraries. And to port from MS-DOS or OS/2 to UNIX, just recompile.

Trial with a smile. C-scape is not only the most sophisticated, flexible and powerful interface system available, it's also the most friendly—and easiest to use. Try C-scape on a 30-day trial. It comes with a thorough manual, demo disk, sample programs with source code, an optional screen designer and code generator, access to a 24-hour bulletin board, and toll-free support. No royalties, runtime licenses, or runtime modules. After you register, you get complete library source code at no extra cost.

Call 800-233-3733 (617-491-7311 in Mass.) to try C-scape now. After the joy of C-scape, programming will never be the same.

MS-DOS, OS/2: $399, library only; with Look & Feel, $499. UNIX, XENIX, Apollo, Sun, Stratus, others please call. Mastercard and Visa accepted.

C-scape Features

Graphics. Combine high-resolution color graphics with text or menus.

Object-oriented. Add features and create reusable code modules.

Mouse. Use any standard mouse for fast screen control.

Portability. Write hardware independent code. Supports DOS, OS/2, UNIX, others. Autodetects Hercules, CGA, EGA, VGA.

Text editing. Create a full-featured text editor or pop-up note pad.

Field flexibility. Create masked, protected and marked fields with complete data validation. Use time, date, money, pop-up list, and many more functions, or create your own.

Windows. Choose from pop-up, tiled, bordered and exploding windows, with size and numbers limited only by RAM.

Menus. Choose from pop-up, pull-down, 123-style, or slug menus, or create your own.

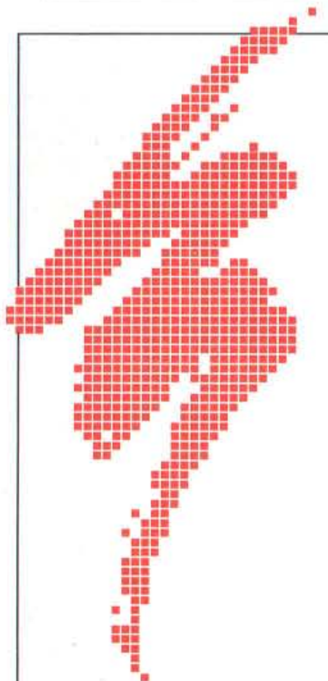
Context-sensitive help. Link help messages to individual screens or fields. Cross reference messages to create hypertext-like help.

Screen design. Build any type of screen or form with the Look and Feel™ Screen Designer, then automatically convert it to C.

Screen flexibility. Call screens from files at run time or link them in.

OAKLAND

Oakland Group, Inc. 675 Massachusetts Ave., Cambridge, MA 02139 USA. FAX: 617-868-4440; **Washington** 206-746-8767; **Benelux** (02159)46814; **Denmark** (02)88 72 49; **France** (1)46 09 28 28; **Germany/Austria/Switzerland** (49)07127/5244; **Norway** (02)44 88 55; **Sweden** (013)124780; **U.K.** (0992)500919. C-scape and Look & Feel are trademarks of Oakland Group, Inc. MS-DOS and XENIX are trademarks of Microsoft Corp. OS/2 is a trademark of International Business Machines Corp. UNIX is a trademark of AT&T. HERCULES is a trademark of Hercules Computer Technology, Inc. Prices and terms subject to change.



GRAPHICS!

Add lightning fast graphics to your programs quickly and easily through the popular PCX file format. Why reinvent the wheel? Make your programs immediately compatible with hundreds of packages from Aldus PageMaker to ZSoft's PC Paintbrush with these linkable graphic libraries.

"An exceptional product" - Programmer's Journal, Aug '89 **$195**

Version 3.5 of the **PCX Programmer's Toolkit** gives you over 60 powerful functions to manipulate bitmapped graphics. Use Virtual screens, Super VGA modes, LIM 4.0 support, a 300 page manual, 9 utilities including screen capture and display, and the fastest routines on the market.

Need Special Effects, but caught in a GRASP? **$99**

Why create a demo when you can create the real thing? Don't be trapped in a slideshow editor or demo program when you can use **PCX Effects** for the PCX Toolkit and your favorite programming language. A Music Language and spectacular effects for exploding your graphics!

Blazing Graphics Text **$149**

With **PCX Text** you can display text with graphics as fast as it always should have been. Display characters, strings, fixed and proportional text, background transparency, and more. Includes a font editor, 85 fonts, and text utilities for blazing graphics bitmapped text.

NO ROYALTIES!
30-DAY MONEY-BACK GUARANTEE!
SOURCE CODE AVAILABLE!

All packages support 12 compilers for C, Pascal, Basic, Fortran, Assembly, and Clipper. All modes of the Hercules, CGA, EGA, VGA, and Super VGA adapters are supported, up through 800x600x256 (22 modes in all). Assembly Language source code is optionally available. Trademarks are property of their respective holders.

G·E·N·U·S
MICROPROGRAMMING
11315 Meadow Lake • Houston, Texas 77077 • (713) 870-0737
FAX: 713-870-0288
Add $5/package for UPS Ground
Texas Residents add 8% Sales Tax

For orders and information, call:

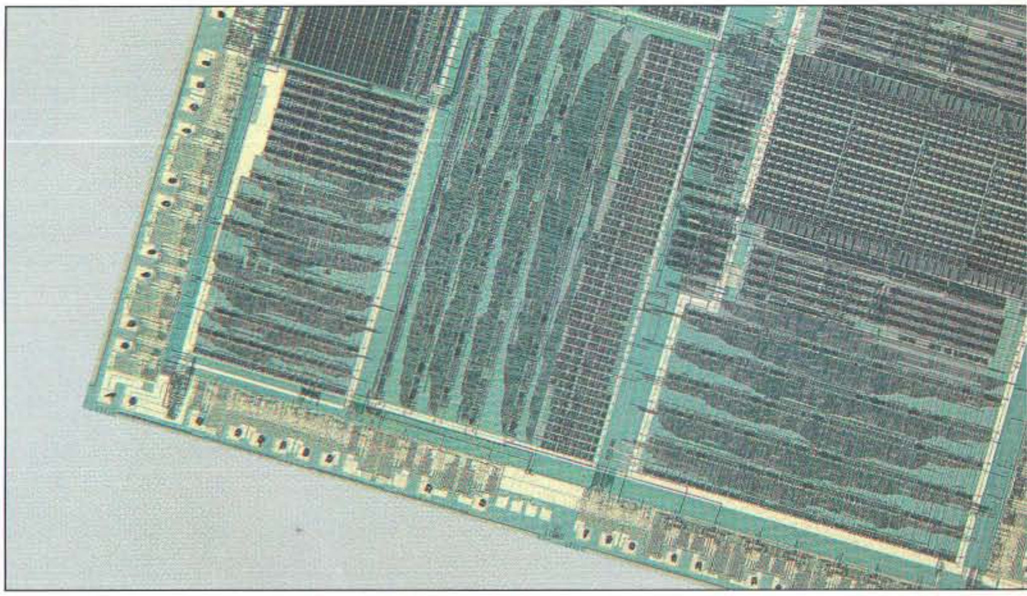
1-800-227-0918

VISA/MC/AMEX/COD/CHECK/PO

VESA VGA BIOS Extensions

A software standard for Super VGA

Bo Ericsson



An integral part of IBM's PS/2 announcement in April 1987 was the video graphics array (VGA) system. Based on the architecture of the enhanced graphics adapter (EGA), the VGA offered extended resolutions and a new 256-color video mode. Since that time, the VGA has grown in importance and is today an established PC video standard. As a matter of fact, all "old" video standards — the monochrome display adapter (MDA), color graphics adapter (CGA), Hercules Graphics Adapter, and EGA — are quickly losing ground to the VGA.

There are several reasons for the VGA's success. For one thing, the new VGA resolutions (see Figure 1), together with lower-priced multi-frequency monitors, have made the VGA a more attractive solution than previous standards. Also, a multitude of VGA offerings and fierce competition have made a baseline VGA an economically attractive choice.

As a matter of fact, competition in the VGA marketplace not only has driven the prices of VGA boards to the bottom, but has pushed up the features and capabilities of these boards. Virtually all VGA controllers available today are compatible down to the register level with the IBM VGA, and almost all of them implement some extensions to the IBM VGA.

The term "Super VGA" is used in this article to identify video hardware that implements a full superset of the standard VGA, including register compatibility. Extensions to the IBM VGA can be classified into three different categories:

1. Backwards compatibility
2. Functional extensions
3. Higher spatial and color resolutions

Backwards Compatibility

The basic IBM VGA is, at best, compatible with older video standards only at the BIOS level. There is a large population of older programs written specifically for, and directly to, the CGA or Hercules Graphics Adapter that bypass the BIOS partially or completely. Because of this, none of these applications run on a standard VGA.

However, most VGA products offer some register-level support for these older standards. These implementations either attempt to automatically detect older programs and switch into a suitable compatible video mode or require a utility program to lock the video hardware into a compatible video mode.

Functional Extensions

The basic VGA is a pretty dumb device; the CPU (that is, the application program) is required to do almost all graphics processing. Only certain logical operations on the graphics data can be performed by the standard VGA hardware. There are no functions for *BitBlts* (bit-block-transfers), line drawing, and so on.

Bo is a software engineering manager at Chips and Technologies Inc., 3050 Zanker Road, San Jose, CA 95134.

In graphics-intensive applications, such as MS-Windows, OS/2 Presentation Manager, and GEM, manipulating the graphics bitmap takes considerable time and affects system performance. For this reason, several VGA controller vendors have put various graphics capabilities directly into the VGA hardware.

For instance, certain VGA controllers implement a graphics cursor in hardware. All graphics user interfaces (such as Windows, GEM, X-Windows, Presentation Manager, etc.) use a graphics cursor. The graphics cursor is an icon (usually an arrow) that moves around the screen as the mouse is moved. A lot of CPU processing is required to move the graphics cursor even one pixel on the screen. Instead of refreshing the actual bitmap on a standard VGA, these controllers need only the coordinate of the "hot-spot." The actual display of the cursor is done in hardware; bitmap manipulation is not necessary.

Other VGA controllers implement more sophisticated write modes, elementary *BitBlt* capabilities, or other functions that relieve the CPU of some graphics processing.

Higher Resolutions, More Colors

The most exciting aspect of all Super VGA implementations, however, is the higher resolutions and the increased number of simultaneous colors on the screen. The standard VGA can display 16 simultaneous colors in 640 × 480 resolution and 256 colors in 320 × 200, as described in Figure 1. In contrast, a typical Super VGA board can do 1024 × 768 in 16 colors and 640 × 480 in 256 colors. In the near future, a range of VGA controllers will be able to do 1024 × 768 in 256 colors. And a little further down the line, some controllers will have the capability of 1280 × 1024 resolution in 16 colors.

Developments in the monitor market make these extended resolutions especially important. Multifrequency monitors capable of resolutions up to 1024 × 768 are available today for less than $1000, and the price is expected to drop even further.

Planar vs. Packed Pixel Modes

Before beginning a discussion on Super VGA graphics, a brief summary of the basic video memory modes is required. VGA graphics video modes use either planar or packed pixel video memory architecture.

In planar mode, the video memory is divided into four separate planes. One pixel is defined by 4 bits, 1 bit per plane. Eight pixels are defined by 4 bytes, 1 byte per plane. Because one pixel is defined by 4 bits, 16 colors can simultaneously be displayed.

Normally, only one plane can be accessed at one time by the CPU. To access another plane, the hardware registers of the VGA have to be reprogrammed. For rapid fills of a large area to a certain color, the VGA can be programmed for 32-bit operation, allowing simultaneous access to all four planes.

In packed pixel mode, only one memory plane is available. One pixel is defined by 1 byte in the memory, yielding 256 simultaneous colors.

The Developer's Dilemma

In spite of this revolution and the fantastic opportunities that Super VGA provides, software development has been slow in tapping into the capabilities. Very few applications have Super VGA support, and only OEM-specific display drivers (software tied directly to a certain VGA controller) can generally exploit Super VGA resolutions and capabilities.

There are several reasons why software development for Super VGA has been sluggish. The most important reason is that almost all Super VGA hardware implementations are different from one another — a Super VGA controller from manufacturer A is usually significantly different from manufacturer B's because no common hardware or software interface exists.

The software developer has to gather a significant understanding of intimate details of each Super VGA controller (of which there are at least ten at present) and each implementation (of which there are dozens, maybe hundreds) that he/she intends to support. The cost of acquiring this knowledge and supporting these disparate environments is prohibitively high; software developers have shunned Super VGA for this reason.

Non-standard Initialization

Super VGA implementations differ significantly in the video mode initialization procedure. One piece of mode setting code will not work on more than one Super VGA board because the I/O addresses for the extended registers required for Super VGA operation vary from implementation to implementation. In addition, the specific parameters for the registers all depend on the VGA controller.

Another aspect of this problem is that there is no uniform BIOS support for mode initialization across Super VGA products. No video mode number scheme exists. A 640 × 480 256 color video mode is called 79 in one implementation and 43 in another. Also, no standardized mode initialization call exists.

All this means that an application cannot program the hardware directly (because no standard hardware exists), nor can it call a BIOS to initialize the mode (because a standardized mode number doesn't exist, and because no standardized calling sequence is established).

Different Windowing Schemes

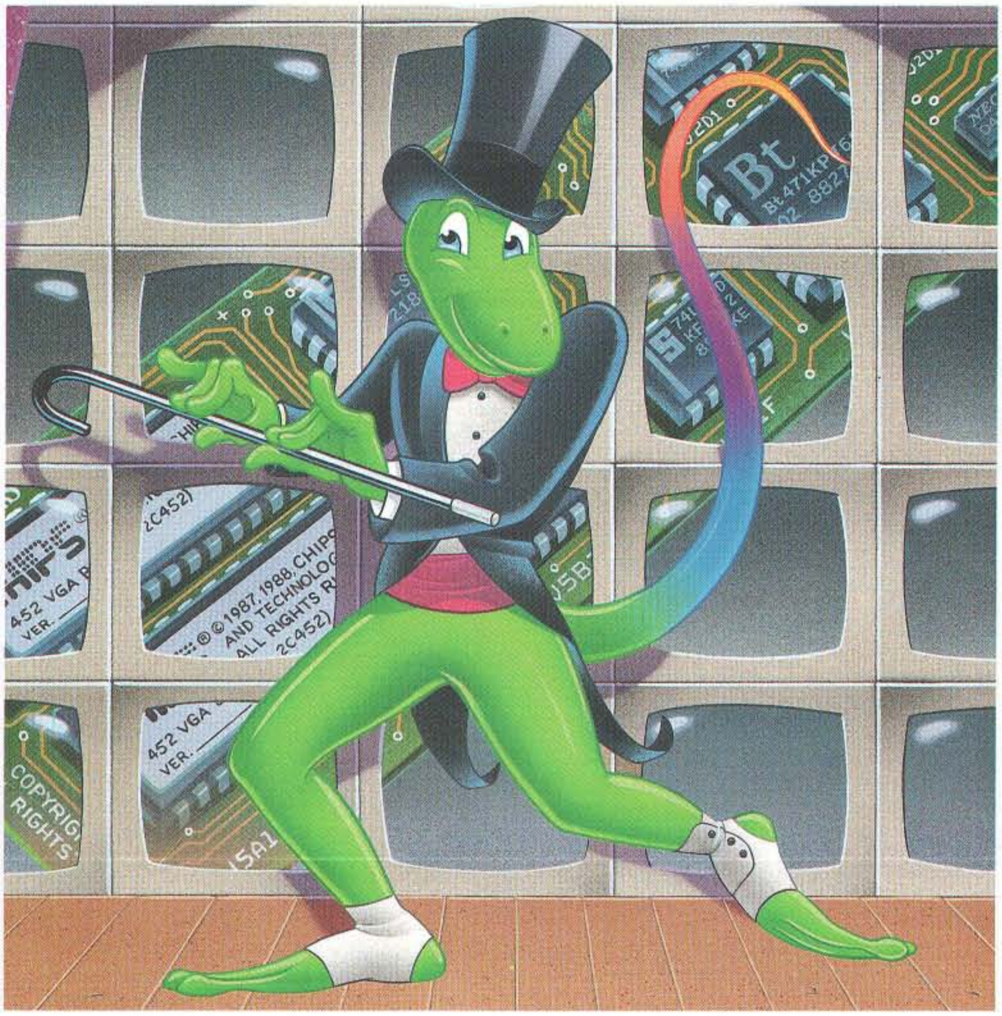
Another area where Super VGA implementations differ greatly is in how the video memory is accessed. In the IBM PC, a maximum of 128K is devoted to the video system. This address space is located between A0000 and BFFFF hex. For compatibility reasons, only the 64K at A0000 is normally used for

	Resolutions					
Colors	320 × 200	640 × 200	640 × 350	640 × 480	800 × 600	1024 × 768
2	CGA	CGA	EGA	VGA	Super VGA	Super VGA
4	CGA	EGA	EGA	VGA	Super VGA	Super VGA
16	EGA	EGA	EGA	VGA	Super VGA	Super VGA
256	VGA	Super VGA	Super VGA	Super VGA	Super VGA	Super VGA

Figure 1: *PC graphics resolutions and colors*

Resolution	Colors	Pixels	Bits per pixel	Total memory (bytes)	Planes	CPU memory (bytes)
640 × 480	16	307200	4	153600	4	38400
800 × 600	16	480000	4	240000	4	60000
1024 × 768	16	786432	4	393216	4	98304
640 × 400	256	256000	8	256000	1	256000
640 × 480	256	307200	8	307200	1	307200
800 × 600	256	480000	8	480000	1	480000
1024 × 768	256	786432	8	786432	1	786432

Figure 2: *Memory requirements of Super VGA modes*



Any VGA card can look like a star until you see it perform alone.

Get the Premier VGA CAD Card—all the extras without extra cost.

Nowadays, it seems like everybody makes a VGA card. But then they charge for software driver updates. Or memory modules. Or specialized VGA add-on cards to fill the gaps where they come up short in CAD or animation performance. Before you know it, you're up to your slot limits in expensive add-ons and workarounds.

It's enough to make you want a card engineered for an engineer, like the Premier VGA CAD Card.

With the Premier VGA CAD Card, you get high-resolution pass through and video-ready capability. Use the *only* direct connector to Magni's VGA Producer.™ Enjoy up to 1024 x 768 non-interlaced support with 512K on-board memory. And download free application drivers for the latest versions of popular software such as Microstation, Microsoft® Windows and AutoCAD® from the Metheus bulletin board. All at less cost than mix-and-matching your own VGA And add-on cards!

Whether you're starting out in CAD and need room to grow, or are pushing the limits of your current CAD system and need a *real* CAD card, Premier VGA is the card for you. Get Premier VGA and set the stage for professional performance. Call Metheus at 1-800-638-4842.

METHEUS

OGC Science Park, 1600 NW Compton Drive, Beaverton, OR 97006-6905

CIRCLE NO. [illegible] ON READER SERVICE CARD

Super VGA resolutions (another video board in the system might be located at B0000-BFFFF).

However, Super VGA video modes consume more video memory than is available in the CPU address space. Figure 2 details typical memory requirements of Super VGA modes. As is evident from this table, there has to be a mechanism for the CPU to reach into the video memory using the 64K (or 128K) "window" available in the CPU address space.

Unfortunately, there are almost as many windowing schemes as there are Super VGA controllers. Some controllers have one window into the video memory, while others have two. Some controllers have separate read and write windows, while others allow read/write in both windows. Some controllers implement a "sliding" windowing scheme, whereby a window can be placed on any boundary in the video memory, while others allow placement of the window only on a 64K boundary.

On top of this, the hardware registers that control the windowing scheme are located at different I/O addresses and require different parameters.

Enter the VESA BIOS Extension

The Super VGA BIOS extension standard, as defined by the Video Electronics Standards Association (VESA), intends to remedy the incompatibility issues addressed earlier. The standard tries to address all major problems a software developer faces when writing software for Super VGA.

Technically, the VESA BIOS extension is implemented as an addition to the regular video BIOS, accessed through software interrupt 10 hex. Standard video BIOS functions are called by placing function numbers in the range from 0 to 1C hex, depending on the function, in the AH CPU register and then generating a software interrupt 10 hex. To call a VESA BIOS function, the application would place the value 4F hex in the AH register, place a function number in the AL register, and then generate an interrupt 10 hex. Figure 3 describes the VESA BIOS extension functions.

The VESA BIOS extension may be placed in ROM together with the regular BIOS. It may also be implemented as a device driver, loaded by the operating system at boot time. Initially, most VESA BIOS extensions will be available as TSR programs. To the application, the method of implementation is irrelevant; functionally, the BIOS extension behaves the same.

The VESA BIOS extension provides two fundamental services to the application program:

1. Information
2. Hardware setup

Global Information

To be able to adapt to a specific Super VGA environment, an application needs several important pieces of information. First and foremost, an application needs to know whether the specific environment is indeed capable of Super VGA resolutions. The application also needs to know whether any VESA support is available. In addition, certain applications might want to identify a specific VGA controller.

This kind of global information is provided by VESA BIOS function 0, *Return Super VGA mode information*. Before the application calls this function, it has to allocate a buffer of 256 bytes. The VESA BIOS extension will fill this buffer with various types of information.

One of the most important pieces of information returned by function *0* is a pointer to a list of Super VGA modes supported by the display adapter. These video modes can be VESA-defined modes as well as OEM-defined modes. See Figure 4 for a list of VESA-defined video modes.

Mode-specific Information

To determine the characteristics of a particular video mode, the application would then call VESA BIOS function 1, *Return Super VGA mode information*. Like function *0*, the application has to allocate a 256-byte buffer prior to making the function call.

On return from the function, the VESA BIOS extension will have filled a struc-

The following functions are defined by the VESA BIOS extension. They are all accessible through interrupt 10 hex with AH set to 4F hex.

Every function returns status information in the AX register. The format of the status word is as follows:

AL== 4Fh:	Function is supported
AL!= 4Fh:	Function is not supported
AH== 00h:	Function call successful
AH== 01h:	Functiion call failed

Function 0 - Return Super VGA Information

```
Input:     AH=4Fh     Super VGA support
           AL=00h     Return Super VGA information
           ES:DI=     Pointer to information block
Output:    AX=        Status
           All other registers are preserved
```

The information block has the following structure:

```
VgaInfoBlock struc
  VESASignature   db   'VESA'     ; 4 signature bytes
  VESAVersion     dw   ?          ; VESA version number
  OEMStringPtr    dd   ?          : Pointer to OEM string
  Capabilities    db   4 dup(?)   ; capabilities of the video environment
  VideoModePtr    dd   ?          ; pointer to supported Super VGA modes
VgaInfoBlock ends
```

Function 1 - Return Super VGA mode information

```
Input:     AH=4Fh     Super VGA support
           AL=01h     Return Super VGA information
           CX=        Super VGA video mode
           ES:DI=     Pointer to information block
Output:    AX=        Status
           All other registers are preserved
```

Function 2 - Set Super VGA video mode

```
Input:     AH=4Fh     Super VGA support
           AL=02h     Set Super VGA video mode
           BX=        D0-D14 = video mode
                      D15 = Clear memory flag
                            0 = Clear video memory
                            1 = Don't clear video memory
Output:    AX=        Status
           All other registers are preserved
```

Function 3 - Return current video mode

```
Input:     AH=4Fh     Super VGA support
           AL=03h     Return current video mode
Output:    AX=        Status
           BX=        Current video mode
                All other registers are preserved
```

Figure 3: *VESA BIOS extension functions (accessible through interrupt 10 hex with AH set to 4F hex)*

ture, called the *ModeInfoBlock*, with all relevant information about this video mode. See Figure 5 for a description of the *ModeInfoBlock*.

Mode Attributes

The first word (16 bits) in the *ModeInfoBlock*, the *ModeAttributes* field, specifies several important characteristics of the video mode. See Figure 6 for the layout of this field.

Bit *D0* in the *ModeAttributes* field specifies whether the mode is supported by the present hardware configuration. If a particular video mode requires a certain monitor, and this monitor is presently not connected to the system, this bit can be cleared to block access to the mode. Applications should never try to initialize a video mode whose *ModeAttributes D0* is set to *0*.

As will be evident in the discussion later, the VESA BIOS function *0* returns a lot of information to the application. Some of this information is mandatory, some is optional. Bit *D1* of the *ModeAttributes* specifies whether any optional information is available.

Bit *D2* indicates whether the output functions (TTY output, set/get pixel, scroll window, etc.) of the regular video BIOS can be used in this video mode. It is not mandatory for a VESA BIOS extension to support all or any output functions in Super VGA modes. The primary reason for this is that high-performance applications handle all output themselves anyway, for performance reasons. The fact that output support consumes a lot of precious memory space in a ROM-based implementation was also important in making this support optional. If bit *D2* is cleared, then no output support is available.

Bit *D3* specifies whether the mode is monochrome (*D3=0*) or color (*D3=1*). Bit *D4* defines the mode as either text mode (*D4=0*) or graphics mode (*D4=1*).

Window Description

The characteristics of the windowing system are described in the next field in the *ModeInfoBlock* structure. The *WinAAttributes* and *WinBAttributes* identify whether window A and B exist and are readable or writeable. All Super VGA boards capable of resolutions beyond 640 × 400 in 256 colors and 800 × 600 in 16 colors have at least one window into the video memory. Applications can determine the existence of a second window by testing bit *D0* of *WinBAttributes*.

The *WinGranularity* identifies the smallest address boundary that the window can be placed upon. In today's Super VGA boards, this varies from 1K to 64K. The *WinSize* field identifies the size of the windows. In a single-window system, the size is normally 64K, while in a dual window system, the size is normally 32K.

The location of the windows within the CPU address space is specified by the fields *WinASegment* and *WinBSegment*. Normally Window A is located at address A0000. If a second window is present, it would typically be located at A8000 or B0000. If the VGA controller implements different read and write windows, the second window could be located at the same CPU address as the first window. In such a system, a CPU read will access the read window, while a CPU write will access the write window.

The *WinFuncAddr* field specifies a direct address to the windowing function (Figure 3, VESA BIOS function 5). The standard way to access the video BIOS and the VESA BIOS extension is to generate an *int 10*. However, due to the large number of subfunctions using *int 10*, function dispatching may take considerable time. This makes *int 10* too slow for some graphics operations. One such time-critical operation is changing the windowing registers. By using the absolute address to the function, an application can issue a *far*

Function 4 - Save/Restore Super VGA video state

Input:	AH=4Fh	Super VGA support
	AL=04h	Save/restore Super VGA video state
	DL=00h	Return save/restore state buffer size
	CX=	Requested states
		D0=Save/restore video hardware state
		D1=Save/restore video BIOS data state
		D2=Save/restore video DAC state
		D3=Save/restore Super VGA state
Output:	AX=	Status
	BX=	Number of 64-byte blocks to hold the state buffer
	All other registers are preserved	
Input:	AH=4Fh	Super VGA support
	AL=04h	Save/Restore Super VGA state
	DL=01h	Save Super VGA video state
	CX=	Requested states (see above)
	ES:BX=	Pointer to buffer
Output:	AX=	Status
	All other registers are preserved	
Input:	AH=4Fh	Super VGA support
	AL=04h	Save/Restore Super VGA state
	DL=02h	Restore Super VGA video state
	CX=	Requested states (see above)
	ES:BX=	Pointer to buffer
Output:	AX=	Status
	All other registers are preserved	

Section 5 - CPU Video Memory Window Control

Input:	AH=4Fh	Super VGA support
	AL=05h	Super VGA video memory window control
	BH=00h	Select Super VGA video memory window
	BL=	Window number
		0=Window A
		1=Window B
	DX=	Window position in video meory
Output:	AX=	Status
Input:	AH=4Fh	Super VGA support
	AL=05h	Super VGA video memory window control
	BH=01h	Return Super VGA video memory window
	BL=	Window number
		0=Window
		1=Window
Output:	AX=	Status
	DX=	Window position in video memory

Mode number	Resolution	Colors
100h	640 × 400	256
101h	640 × 480	256
102h	800 × 600	16
103h	800 × 600	256
104h	1024 × 768	16
105h	1024 × 768	256
106h	1280 × 1024	16
100h	1280 × 1024	256

Figure 4: *VESA-defined Super VGA modes*

call directly into it, speeding up execution considerably.

Optional Information

Only a portion of the *ModeInfoBlock* is obligatory information. The other section is optional and is provided if the specific mode is nonstandard. None of the modes defined by VESA (see Figure 4) require the optional information. For an OEM-specific mode, however, the VESA BIOS extension needs to inform the application about items such as screen resolution, number of planes, and bits per pixel.

Refer to the VESA BIOS extension specification for information on how to use these optional fields.

Video Mode Initialization

One main objective of the VESA BIOS extension is to help applications set up video modes. This is realized through VESA BIOS Function 2, *Set Super VGA video mode.* The application simply places the video mode to be initialized in the BX register and calls this function. Normally, the video memory will be cleared, but if the application sets bit *D15* of the BX register prior to calling the function, the memory will be preserved.

VESA mode numbers are 15 bits wide (see Figure 4). OEM-defined mode numbers are 7-bits wide and are implemented as a subset of VESA-defined modes. Due to this numbering convention, VESA modes, OEM-specific modes, and regular VGA modes can be initialized by using VESA BIOS Function 2.

If an application needs to know the present video mode, it would call VESA BIOS Function 3, *Return current video mode.* For applications (especially TSR programs) that need to interrupt other programs, the VESA BIOS Function 4, *Save/Restore Super VGA video state,* comes in handy.

The Windowing Function

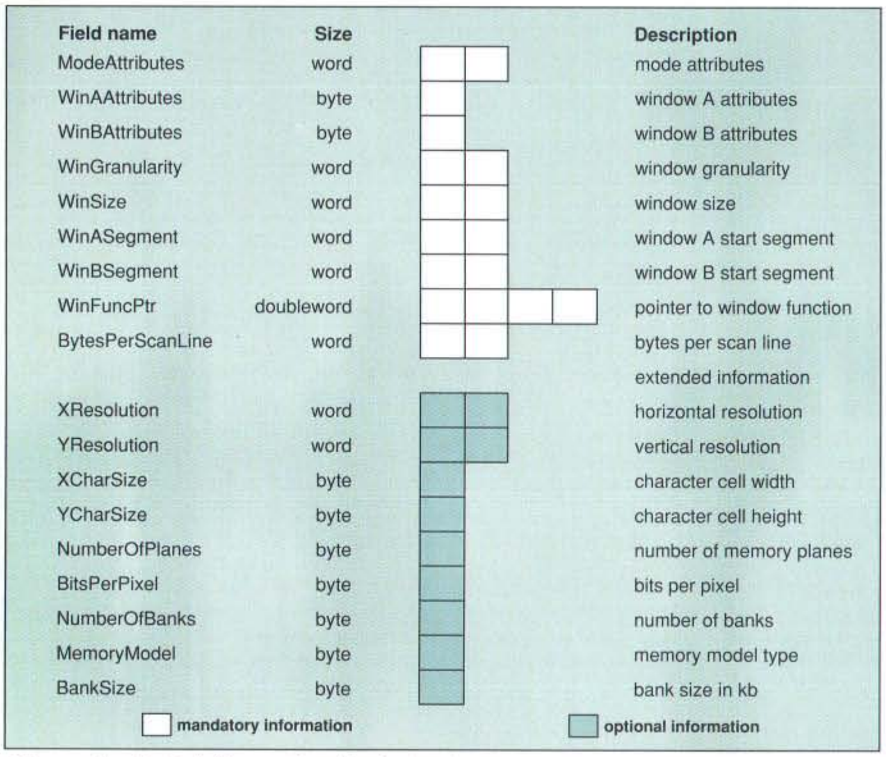
Finally, the VESA BIOS extension provides a mechanism to control the position of the video memory windows. This is handled by Function 5, *CPU video memory window control.* To reposition a window into the video memory, the application simply places the window position in the DX register, the window number (*0* for Window A and *1* for Window B) in the BL register, and calls Function 5.

The window position is not specified as a byte offset, but rather in terms of granularity units. As stated earlier, the window granularity expresses the smallest boundary on which the window can be placed. Today's Super VGA boards have granularities between 4K and 64K. Thus, if the granularity is 16K, and the application wants to position the window at 64K, the window position is 64/16 = 4 granularity units.

Conclusion

The VESA BIOS extension provides all necessary information and programming support to Super VGA applications. For the first time, it is possible to develop generic graphics software, tapping into the exciting capabilities of Super VGA.

However, just because the VESA BIOS extension has made it possible to write such applications doesn't mean it will be trivial. Most of the complexity in dealing with Super VGA stems from managing windows into the video memory. Anyone already familiar with writing software for one Super VGA board should have no difficulty in programming others using the VESA BIOS extension.



Field name	Size		Description
ModeAttributes	word	☐☐	mode attributes
WinAAttributes	byte	☐	window A attributes
WinBAttributes	byte	☐	window B attributes
WinGranularity	word	☐☐	window granularity
WinSize	word	☐☐	window size
WinASegment	word	☐☐	window A start segment
WinBSegment	word	☐☐	window B start segment
WinFuncPtr	doubleword	☐☐☐☐	pointer to window function
BytesPerScanLine	word	☐☐	bytes per scan line
			extended information
XResolution	word	▨▨	horizontal resolution
YResolution	word	▨▨	vertical resolution
XCharSize	byte	▨	character cell width
YCharSize	byte	▨	character cell height
NumberOfPlanes	byte	▨	number of memory planes
BitsPerPixel	byte	▨	bits per pixel
NumberOfBanks	byte	▨	number of banks
MemoryModel	byte	▨	memory model type
BankSize	byte	▨	bank size in kb

☐ mandatory information ▨ optional information

Figure 5: *Mode information block structure*

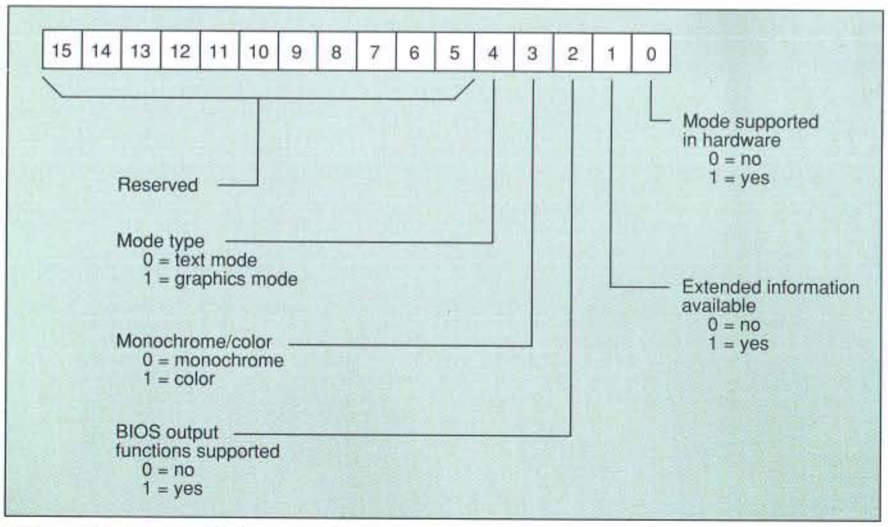


Figure 6: *Mode attribute field*

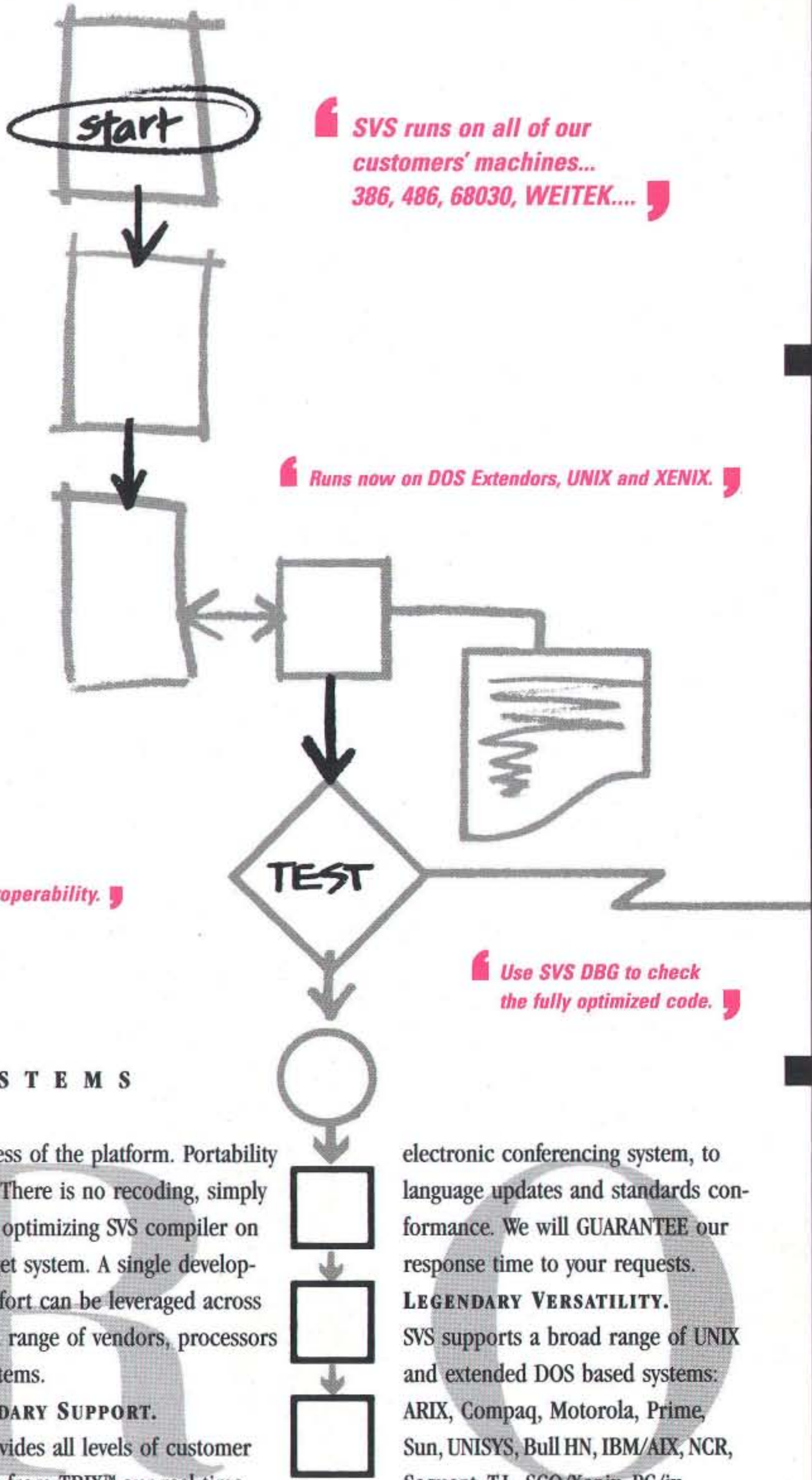
For More Information

Video Electronics Standard
1330 S. Bascom Ave., Ste. D
San Jose, CA 95128-4502
408-971-7525

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 12.**

SVS - LEGENDARY COMPILERS. THE PERFORMANCE BENCHMARK.



“SVS runs on all of our customers' machines... 386, 486, 68030, WEITEK....”

“SVS benchmarked fastest in MIPS magazine.”

“Runs now on DOS Extendors, UNIX and XENIX.”

“Call SVS for optimizing compiler. 415-572-8800”

“Call FORTRAN math routines from C...great interoperability.”

“Use SVS DBG to check the fully optimized code.”

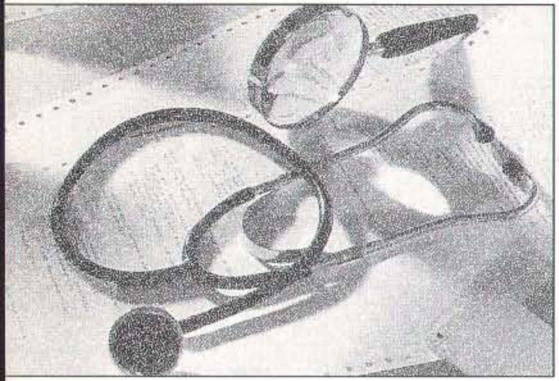
SVS LANGUAGE SYSTEMS

LEGENDARY COMPILERS.
SVS C, SVS Fortran, SVS PASCAL, SVS Basic-PLUS, SVS DBG.
LEGENDARY SYSTEMS.
In worldwide distribution for over ten years, SVS Language Systems are licensed at over 250,000 UNIX™ sites.
LEGENDARY PORTABILITY.
Applications using SVS Language Systems use identical source code regardless of the platform. Portability is easy. There is no recoding, simply use the optimizing SVS compiler on the target system. A single development effort can be leveraged across a broad range of vendors, processors and systems.
LEGENDARY SUPPORT.
SVS provides all levels of customer support from TRIX™ our real-time electronic conferencing system, to language updates and standards conformance. We will GUARANTEE our response time to your requests.
LEGENDARY VERSATILITY.
SVS supports a broad range of UNIX and extended DOS based systems: ARIX, Compaq, Motorola, Prime, Sun, UNISYS, Bull HN, IBM/AIX, NCR, Sequent, T.I., SCO/Xenix, PC/ix.

SILICON VALLEY SOFTWARE
TRIO SOFTWARE SYSTEMS CORPORATION

1710 South Amphlett Blvd., Suite 100, San Mateo, CA 94402 415-572-8800

CIRCLE NO. 46 ON READER SERVICE CARD



Cruising with TopSpeed

A full-featured toolset is the real value in this C compiler

Alex Lane

It is fashionable in some circles to yawn upon hearing that a new C compiler has hit the market. Such folks see the C world as divided into two main camps — the Microsofts and the Turbos — with a smattering of fanatics representing an insignificant fringe. That fringe, however, has lately succeeded in whipping up the C market, the result being a lively free-for-all as new arrivals such as Watcom, Zortech, and now TopSpeed C attempt to prove their worth to programmers. Readers, familiar with the TopSpeed name, will associate it with a popular Modula-2 compiler and a recently introduced Pascal compiler, marketed by Jensen & Partners International. JPI, a company established in 1987 by a group of former Borland employees, is a relatively small company with unquestionably large vision. They intend to develop an integrated multilanguage environment that will let programmers seamlessly mix and match routines from a broad spectrum of languages, including ISO Pascal, C, C++, Modula-2, and Ada. What JPI in effect proposes to do is to link each language dynamically into the system as an overlay at run time, thus allowing each language compiler to use the same optimizing code generator. If TopSpeed C is any indicator, JPI has its sights set on a worthwhile goal.

Alex is a knowledge engineer for Technology Applications Inc. in Jacksonville, Florida. He can be reached on BIX as a.lane or through MCI mail as ALANE.

What You Get

I reviewed the TopSpeed C, Version 1.02, Extended Edition, which is basically the standard package (comprised of an optimizing 100 percent ANSI C compiler and high-speed linker, an automatic *make* facility, an editing environment, and source-level debugger) combined with the TopSpeed C TechKit, which provides enhanced functionality in the form of library source code, Windows support, DOS dynamic linking, profiling, and post-mortem debugging, among other features. The standard TopSpeed package consists of seven diskettes and nearly three inches of paperback documentation, consisting of a user manual, a language reference, a library reference, and a language tutorial. The TechKit comes on an additional four disks and has separate documentation.

Finding Files

I dread installing software that needs to use DOS environment variables to find files on my disk. For one thing, I only have so much DOS environment space; for another, I often find that two different packages use the same DOS environment variable name in different ways, and that no amount of fiddling with SET statements shall allow the twain to meet on a consistent basis. If you have two or more C compilers installed on your hard disk, you probably know what I mean.

If you want TopSpeed C to use DOS environment variables, you can so specify by using the */y* flag from the command line, but why bother? I took an immediate liking to TopSpeed's redirection file feature, which acts as a sort of private environment. A sample redirection file, TS.RED, is reproduced in Figure 1. The syntax is similar to that of DOS paths. The first line indicates that all KABOOOOM.* files are found in the directory C:\ KABOOOOM. Analogously, all other *.C files may be found either in the current directory (denoted by the '.'), or in C:\TS \EXAMPLES, C:\TS\ SRC, or in D:\ FRAC \PROGS. The remaining lines are self-explanatory, except perhaps for the line that refers to *.A files that are TopSpeed assembler files.

By editing TS.REG, then automati-

```
KABOOOOM.* = C:\KABOOOOM
*.C        = .;    C:\TS\EXAMPLES; C:\TS\SRC; D:\FRAC\PROGS;
*.PRJ      = .;    C:\TS\PRJ; C:\TS\EXAMPLES;
*.H        = .;    C:\TS\INCLUDE;C:\TS\EXAMPLES; D:\FRAC\PROGS;
*.A        = .;    C:\TS\LIB; C:\TS\EXAMPLES; C:\TS\SRC

*.OBJ      =       C:\TS\OBJ; C:\TS\LIB;
*.LIB      = .;    C:\TS\LIB; D:\FRAC\PROGS;
*.DOC      = .;    C:\TS\DOC;
TS.RED     = .;    C:\TS\SYS;
```

Figure 1: *A sample redirection file*

cally saving and reloading it, you can change the file search (and storage) behavior of the environment on-the-fly. If, despite your best intentions, you repeatedly end up with all your *.c, *.obj, *.h, and *.exe files in one giant directory, this feature is for you.

Integrated Development Environment

The cornerstone of the integrated environment is, of course, the editor. Out of the box, TopSpeed's editor is configured to use the WordStar command set. You can change part or all of that by editing a configuration file.

Actually, the TopSpeed configuration file TSCFG.TXT affords the user quite a bit of control over not just the editor but the TopSpeed environment in general. The file is a 33K ASCII text file that defines the menu structure, every menu option, the editor commands, and even compilation error messages used by the TopSpeed system. This is the file you edit if you want to make the environment editor act more like, say, Brief than WordStar. A disk file explains how to make the changes, and after only a few minutes, I was able to change the main menu format from vertical to horizontal and to define the Ctrl-F10 keychord as a way to directly access the optimization menu. While there are many things you can change, there are others (such as an inability to extend the undo feature past one event) you have no control over. Once you've finished making changes, you can incorporate them into the TopSpeed environment by running the TSCFG.EXE program.

Although the editor handles up to ten windows (0 – 9), you'd normally edit in windows #1 through #9, because window #0 is a special window, called the "error editor window." This window comes into play after errors and warnings are found during compilation of a source file. The flawed file is displayed in this window with the cursor positioned at the first error, and the corresponding error message appears at the bottom of the window. Pressing F8 moves you forward to the location of the next error; F7, back to the previous one.

When you exit from the TopSpeed environment, the system remembers the contents and status of each window and reloads the same files the next time you enter the environment. You can alternatively start with a "clean slate" by supplying the */n* option on the command line. Another nice touch is a prompt reminding you to save your work when you call up the source-level visual interactive debugger (VID) from the TopSpeed menu.

There are a number of useful options available under the Utilities menu, including an ASCII table, a programmer's calculator that works in decimal, hex, or binary, and a window that lets you see the scan codes for keyboard keys. There is a multiple-file string search capability that works a bit such as *grep*, albeit without the powerful regular expression capabilities of that Unix utility. Other options include the ability to print files, to view files as data (that is, in hex), and to display system information. "System Info" shows the current date, time, and directory, the names of the files being edited in the TopSpeed windows, and a summary of free space on all disks. Be prepared to wait for this report if you have a CD-ROM disk attached to your system, for even though there is no "free space" available on a CD-ROM, there is no way to tell TopSpeed to ignore the drive, which gets interrogated in turn along with the other drives in your system.

There are a number of other features I found useful in this environment, too many in fact to list. Particularly noteworthy to me, however, are the ability to have up to nine generations of backup copies (I have mine set to 3), and the ability to record, load, save, and playback keystroke macros. The instructions for recording a macro were clear, and it took only a couple of minutes for me to create a macro that toggled all optimization on and off. About the only criticism I have of the editing environment is the lack of mouse support and the lack of Unix-style regular expression parsing (as in Brief, for example), but those are relatively minor annoyances.

Compiling

While the editor and its features form an important part of the TopSpeed C package, you can't forget that this is, after all, a C compiler, and the worth of the package ultimately hinges on how well it compiles code.

To help put TopSpeed C through its paces, I worked with a file that had been written in Microsoft C for a fairly simple-minded game of deduction. The playing "field" for this game is a 9 × 15 grid that contains some number of hidden mines, and the player uses the numeric keypad to "walk" a happy-face character through this mine field to a goal. To give the player a fighting chance, the number of mines in adjacent squares is displayed as the player moves from square to square, thus allowing the player to deduce the location of the mines without "stepping" on them. To make life easier, the location of mines may be marked by pressing M and an appropriate key on the numeric keypad. In addition, if an */s* parameter is passed on the command line when the program is invoked, the program won't let the player step on such marked squares. Finally, typing **?** at the start of the game causes the program to start playing by itself until it either gets to the goal or is unable to proceed further through the grid. Aside from being fun to play, the file KABOOOOM.C (see Listing One, page 109) is just under 1000 lines in length and offers a substantial chunk of source code for the compiler to process.

The first attempt to compile the code resulted in a couple of errors. The first error, in the function *DisplayCell*, had to do with a failure to read an embedded ASCII 2 (the happy-face) in the source code. The TopSpeed editor did not read this character, resulting in the assignment *Char = '';* and a subsequent error. Changing the line to *Char = 2;* fixed the problem.

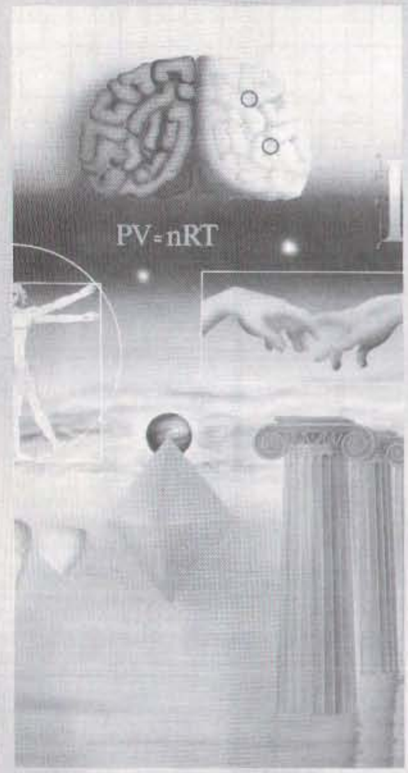
The second error came in a line of the *DisplayChar* function, which read: *FP_SEG(cPos) = 0x0b000;*

While a legitimate statement in Microsoft C, this use of FP_SEG( ) generated a "left operand of assignment must be a modifiable lvalue" error message from the TopSpeed compiler. Pressing F1 for help brought up an explanation of the message, which is comfortably verbose as it is. I moved off the offending line and again pressed F1, and shortly was reading the help screen associated with FP_SEG( ). It referred me to MK_FP( ), which permitted me to replace the offending line (and the line after it) with: *cPos = MK_FP(0x0b800,((x+y*80) << 1));* which eliminated the problem. A quick check showed that Turbo C 2.0 also required the MK_FP( ) syntax in order to compile without error. I later learned that the FP_SEG macro is defined differently for the Microsoft and TopSpeed compilers, which explains the failure to compile.

Once the bugs were corrected, the initial compilation pass with TopSpeed C took about 13 seconds on my 16-MHz ARC 386i computer, and optimization took another 18 seconds or so. With all optimizations turned off (there are nine forms of optimization, including optimization for time and space as well as constant, jump, peephole, loop, and alias optimization), the compile time was cut down to 28 seconds overall. This compared favorably with a run through Microsoft C 5.1, which took 37 seconds to compile without optimization, yet was slower than Turbo C 2.0, which compiled and linked the program in about 13 seconds.

ANSPEC

An Actor Oriented Environment for PDP Specification



ANSpec is a development environment for parallel distributed processing (PDP) models including artificial neural systems. ANSpec is based on recent developments in computer science including actors, objects, polymorphism, operator overloading, and applicative programming. With ANSpec you can specify connectionist systems which capture the full spectrum of inherent concurrency in an application from coarse grained to ultra-fine grained processing architectures.

Platforms supported for ANSpec include PC/AT compatibles with extended memory, MAC-II, IIx, SE, Plus and Sun/3,4, 386i, Sparc and SAIC's Delta II Floating Point Processor.

Introductory prices through July 31, 1990

SAIC Science Applications International Corporation
An Employee-Owned Company

Artificial Neural Systems Division
10260 Campus Point Drive, MS71
San Diego, CA 92121
(619) 546-6290 (FAX) 546-6777

CIRCLE NO. 425 ON READER SERVICE CARD

I ran the compiler from within the editing environment, but you can also run TopSpeed C from the command line, where a comprehensive set of command-line options give the programer complete control of compilation and linking. In fact, there are four ways to set compiler options in the TopSpeed C compiler: From a menu, from the command line, via directives in TopSpeed's *make* facility, and by including pragmas in the source code.

One very topical option in the compiler is the ability to check for ANSI compatibility. JPI has made a point of maintaining 100 percent conformance to the ANSI C definition, and now that the seemingly interminable deliberations of the ANSI C committee have apparently come to a close, this feature should be a point in TopSpeed C's favor.

Making it With Project Files

TopSpeed's project files make it easy to admit to hating traditional make programs. Like its traditional counterpart, TopSpeed's Make uses a text file (called a "project file") to figure out what kind of file to produce with what objects and libraries, and using which memory model. Project files are collections of "directives" that, in addition to the usual specification of object modules, libraries, and so on, establish various compiler and linker options (such as inclusion of debugger information in the .EXE file), override options for specific files or groups of files, and specify what programs to run (if any) after the make process is complete. You could, for example, copy the .EXE file to a diskette every time you compiled and linked the source code. In short, the project file is the mechanism by which TopSpeed source code is transformed into executable files.

Two valuable features aid in the link process. Type-safe linking involves catching function calls made with the wrong parameter types. You will bless this feature the first time it saves you from calling an external function with the wrong parameter types. The technique of smart linking helps keep executable file size down and reduces the complexity associated with maintaining libraries. When you link a program, TopSpeed will only include those routines that are referenced in the code, leaving all the other routines out of the executable file. Strangely enough, this means that sometimes you must make an extraneous reference to a variable in order to make sure certain routines are linked into the .EXE file. A case in point is the need to include a line *includePMD = 1;* in one of the functions that handles critical program errors so that TopSpeed loads the appropriate routines to perform a post-mortem dump in case the program bombs.

Debugging

The VID is a full source-level symbolic debugger that uses overlapping windows in an interactive environment. Like the parent TopSpeed environment, there is no mouse support in VID.

VID is easy to run from the TopSpeed environment, which wisely prompts you to save your files before it swaps itself to disk, leaving room for your program and VID. To use VID, you need to generate VID information during compilation and a .MAP file during the link process. All required files are found using the redirection file, if necessary.

All the usual debugging features are here. You can set and clear breakpoints, create "sticky" breakpoints, examine different types of variables, find procedures, evaluate expressions, all the usual stuff. While not as powerful as, say, the Borland Turbo Debugger (there is no equivalent to the Inspect command, for example, which shows record structure, or the CPU window, which shows registers, memory, and disassembled code all at once) the VID is nevertheless a competent piece of software that is able to do the job.

TechKit

The TechKit is what distinguishes the Extended Edition ($395 list price) from the Standard Edition ($195) of TopSpeed C. What you get for your money is a collection of programs, files, and utilities that add functionality to TopSpeed C. It includes support for Windows programming and dynamic link libraries (DLLs), including DLLs that can be used under DOS. (A DLL is an OS/2 innovation that allows applications to share common data and code by linking library routines at run time.)

A major piece of the Techkit is the source code to the TopSpeed libraries. This collection of files fills over 1.5 Mbytes of disk space. The code is designed not only for use by TopSpeed C, but also for use with other TopSpeed languages. Many of the files are written in TopSpeed's assembler language, which makes for speedy routines, but also requires you to learn a new dialect of assembler.

The TopSpeed Assembler attempts to gain in simplicity and speed by deviating from "standard" 8086 assemblers in several ways. For example, the lexical structure of the assembler is derived from Modula-2, memory operands and segment overrides must always be explicitly stated, and there are no macros used in the language. While

I can understand JPI's reason for doing it this way, I don't look forward to becoming familiar with yet another assembler scheme.

An interesting utility included in the TechKit is WATCH, which lets you specify groups or individual DOS functions to monitor during program execution. I ran the program and specified the date/time functions for monitoring, with output to be sent to my printer (as opposed to the screen or disk file). When I ran KABOOOOM.EXE, a brief report was sent to the printer when the program called DOS to get the time during the initialization phase. The Alt-backspace keychord toggles WATCH on and off, and in order to change the scope of the DOS functions monitored, I found it necessary to unload WATCH and then reload it from scratch. (If you send WATCH's output to the screen, however, you are able to interact more with the program [setting and clearing functions to monitor], albeit at the expense of interfering greatly with the screen.) WATCH, of course, will not work if the program it is monitoring does not use DOS to accomplish its ends.

Other pieces of the TechKit aid in the debug and streamline process. The post-mortem debugger was undoubtedly created for those who've wished their bug-ridden programs could leave some indication behind them of what went wrong before they exit to never-never land. This feature is set by including the PMD.H file and referencing the *includePMD* variable in the source code. Should anything go wrong and a critical error function is called, your program creates a file that details the state of the system just before lights out. This file can be examined using the VID.

I've always gritted my teeth when sitting down to work with a profiler, but I found the TopSpeed TSPROF profiler easy to use. All you need to use TSPROF is a .MAP file, which is created when the program is made. I ran the profiler for KABOOOOM and found that over half the program's time is spent executing DOS routines, nearly half the program's time is spent in the BIOS, and only three percent or so of the time is in the code.

Conclusion

Working with the TopSpeed C compiler was a wholly pleasant experience. The advantage of having the file redirection and project file features are alone almost worth the price of admission, and the overall flexibility of the system is a big plus. Though it remains to be seen whether JPI will be able to successfully market the idea of a common programming environment with plug-in language modules, TopSpeed C certainly deserves to be a contender in the fight for a share of the C compiler market.

Acknowledgments

The author would like to thank Thomas D. Eldredge II for the use of his source code for KABOOOOM.C. Tom's program represents an enhancement of a game called RELENTLESS LOGIC by Conway, Hong, and Smith, which was found on the RBBS-IN-A-BOX CD-ROM.

PRODUCT INFORMATION

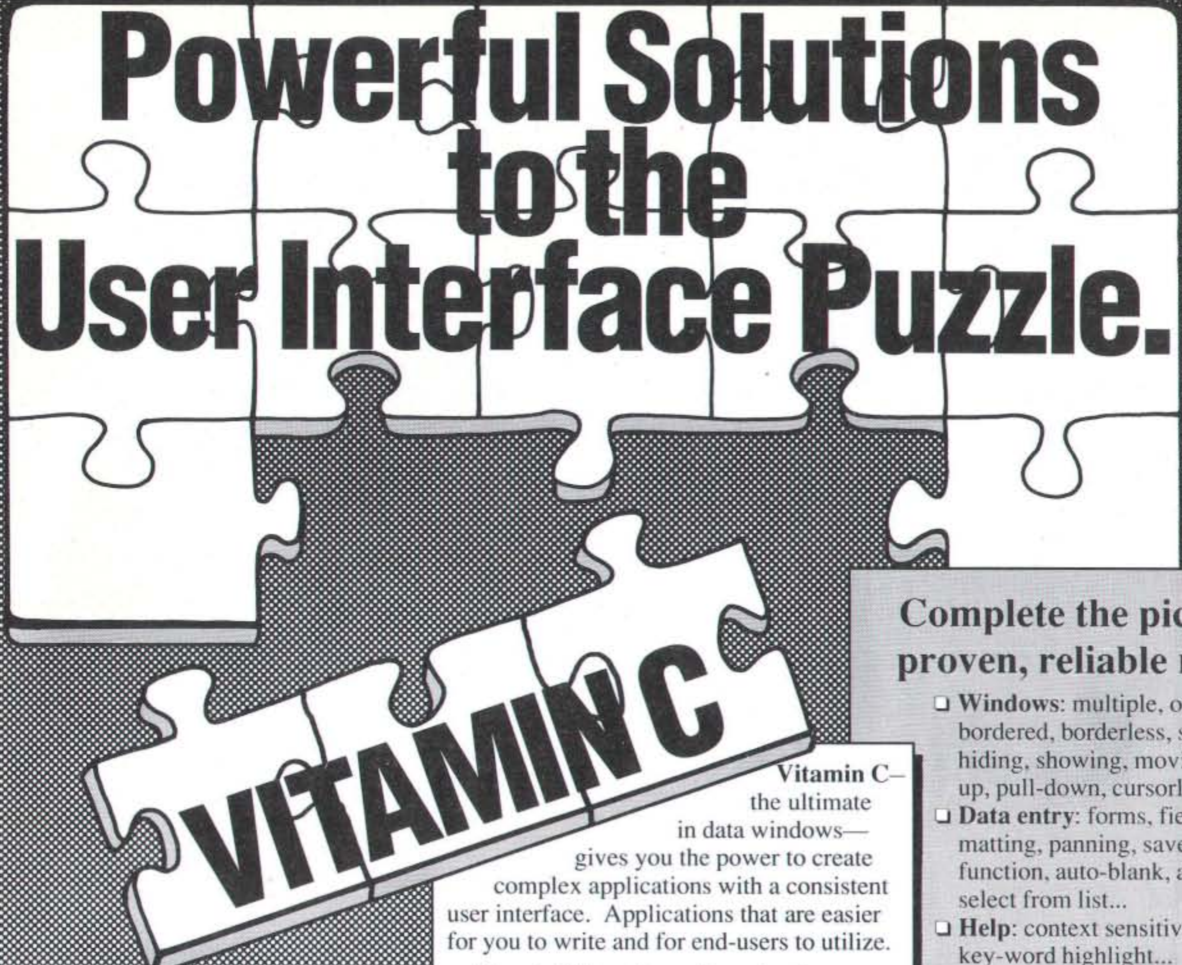
TopSpeed C
Jensen & Partners International
1101 San Antonio Rd., Ste. 301
Mountain View, CA 94043
Price:
Standard Edition $199
Extended Edition $395
OS/2 Edition $495
Requirements: Extended Edition – IBM PC or compatible, DOS 2.0 or later, 640K RAM. Hard disk recommended.

DDJ

(Listing begins on page 109.)

Vote for your favorite feature/article.
Circle Reader Service **No. 7.**

Powerful Solutions to the User Interface Puzzle...



Vitamin C–the ultimate in data windows—gives you the power to create complex applications with a consistent user interface. Applications that are easier for you to write and for end-users to utilize.

Simple C function calls unleash Vitamin C's powerful line-up. Windows, data entry, menus, help—it's all here—no extra modules to buy. Library source and free tech support are included—standard!

Veteran or beginner, you'll get maximum power with minimum effort from Vitamin C's high level functions. Lower level routines offer full control and make it easy to refine your design.

Unrivaled versatility makes you more productive and more competitive. In fact, Vitamin C's open-ended design provides virtually limitless customization potential—even without modifying the source. Plus, Vitamin C is the perfect front-end for your DBMS or other function library.

Advanced features like programmer defined functions before/on-exit/after-exit for each field/menu/menu-item complete the picture.

Top it off with comprehensive printed documentation, no royalties, and our 30-day money back guarantee and you have more than a function library—you have solutions—you have Vitamin C!

CREATIVE PROGRAMMING

Complete the picture with proven, reliable resources.

- ❑ **Windows**: multiple, overlapping, virtual, bordered, borderless, scrolling, printing, hiding, showing, moving, zooming, pop-up, pull-down, cursorless, scrollbars...
- ❑ **Data entry**: forms, fields, validation, formatting, panning, save, abort, pre/post-function, auto-blank, auto-advance, select from list...
- ❑ **Help**: context sensitive, pop-up, default, key-word highlight...
- ❑ **Menus**: multi-level, Lotus, Mac, separators, blank items, unavailable items...
- ❑ **Keyboard**: function keys, jump to function, translation, programmable handler, keyboard idle function...
- ❑ **Text editor**: notepad, word wrap, justify, search, insert, delete, page-up, page-down...
- ❑ **Video**: auto-detect, MDA, CGA, EGA, VGA, 25/43/50 line, programmer definable...
- ❑ **General**: environment variables, command line arguments, date/time math...
- ❑ **Library source** included free!
- ❑ **Free support** via phone and BBS
- ❑ **Portable**: versions for DOS, OS/2, UNIX, Xenix, VMS
- ❑ **Options**: VCScreen–interactive window, form, menu, code generator. Vitamin C on-line documentation.

Call Today!

214•416•6447

Box 112097 Carrollton, Texas 75011

Vitamin C: $225, DOS; $345, OS/2; Call for Unix, Xenix, VMS. **Vitamin C On-line Reference:** $50. Requires the Norton Guides Engine. **VCScreen:** $149. Shipping: Ground, $3; 2-day air, $6; Overnight, $20; Foreign, $30. Visa, MasterCard, American Express. All funds must be U.S. funds drawn on a U.S. bank.

Neural Networks and Image Processing

Finding edges only a human can see

Casimir C. "Casey" Klimasauskas

One of the key problems facing the machine vision industry is how to detect specific features in an image. It turns out that even finding a simple feature such as an edge can be difficult, if not impossible. Even though a person looking at a video camera image on a monitor can readily see the boundary between two objects, it may not be so easy to find it with an algorithm. Researchers studying how the eye preprocesses information for the brain use the term "early vision" for the function of the eye that assists in pattern recognition. We can use insights from research in early vision to solve the problem of edge detection by computer.

This article presents an engineering approximation of early vision, written from the perspective of an engineer investigating useful applications of neurally inspired technology. Although the techniques discussed here were suggested by the processes of the human eye, they are not intended to be biologically accurate, nor is the solution intended to be biologically plausible. The architecture of the edge detection system presented here is the empirical result of exploring many blind alleys and dead ends. For this reason, some of the assumptions and function values used here may seem somewhat arbitrary. Their only justification is that they worked.

The edge enhancement system presented here can be implemented in various ways, using different technologies. This article presents two implementations in software (one using the C language and the other using Lotus 1-2-3) and also describes a third implementation using commercially available image processing hardware and software.

Casimir C. "Casey" Klimasauskas is the founder of NeuralWare Inc., a supplier of neural-network development systems and services. Prior to that he worked extensively in machine vision and robotics. He can be reached at Penn Center West IV-227, Pittsburgh, PA 15276; 412-787-8222.

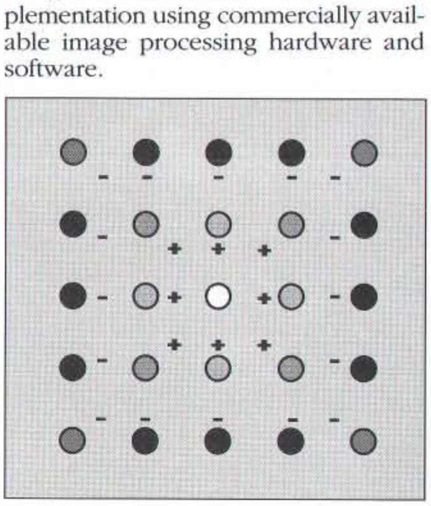


Figure 1: *Effect of a processing element on its neighbors. A "+" near a processing element indicates that the center processing element in the diagram will excite it if it is excited. A "–" near a processing element indicates it will be inhibited if the center processing element is excited*

A Logical Edge Enhancement Model

You might think of the receptive surface of the eye as an array or grid of photoreceptive elements. Light from the outside world impinges on this photoreceptive array and provokes output from each of the array elements. The output of each of these photoreceptors is passed on to another layer of corresponding neurons that work together to enhance the image.

For purposes of this article, we will call our two-layer network the edge enhancement system (EES). Figure 1 shows the effect of one of the EES processing elements. The connections are shown only from the processing element in the center of the array. This processing element excites its nearest neighbors (shown by "+" near the processing elements) and inhibits those a little further away (shown by " – " near the processing elements). The actual strength of the excitation or inhibition, as a function of distance from the center, is shown in Figure 2. When plotted in three-dimensions, with the magnitude of the excitation or inhibition as the Z-axis, the resulting shape looks like a Mexican hat. For this reason, it is sometimes called a "Mexican hat function" (MHF) or "on-center off-surround." The effect of the Mexican hat function is similar to that of a standard image processing filter known as a "difference of Gaussians."

The connections are shown only for the center processing element in Figure 1, all the other processing elements are connected in a similar fashion.

The EES processing element (shown in Figure 3) computes an internal activation value by computing the weighted sum of the outputs of its neighbors and the weights connecting them. This internal activation value is then transformed by a nonlinear transfer function (such as the clamped linear one shown) to produce an actual output. The clamped linear transfer function was found to work best after sigmoid and hyperbolic tangent transfer functions were tried and found not to work. Notice that the current output of a processing element is fed back onto itself as part of the input for computing its internal activation.

Readers familiar with neural-network types will recognize the EES array of processing elements described as a kind of feedback neural net, (similar to a Hopfield network, but with a fixed pattern of inter-connections). The connections are such that each processing element is trying to decide if it is on an edge or not. When this constraint is satisfied, the processing elements reach a stable output state.

In operation, the outputs of the receptor array are passed on to the EES. The initial values of each of the elements in the EES are equal to their corresponding values in the receptor array. After initialization, the EES goes through several iterations. During each iteration the processing elements obtain inputs from their neighbors (either excitatory or inhibitory) as well as from their current state. From these inputs, they compute a new output transformed through some nonlinear function. In the eye, these processes evolve as a dynamical system obeying a set of continuous differential equations defined by the synapses connecting them.

An EES Engineering Approximation

To develop a good engineering approximation, we need to be able to implement the EES inexpensively and efficiently. This section looks at techniques for accomplishing this with readily available off-the-shelf image processing hardware and software. The two principal image processing techniques discussed here are convolution and look-up tables.

Convolution is a common and powerful technique for filtering images. Very simply, a convolution is a specially designed matrix (or filter) that is combined together with a portion of an image to compute a transformed pixel value. The filter is centered at each pixel in the initial image and the "convolution" of the filter and the image beneath it is computed. The result is the transformed value of the center pixel. The matrix is then moved one pixel to the right and the transformed value of the next pixel is computed. When the filter has been applied, centered at each pixel in the initial image, the resulting transformed image is complete. This is shown in Figure 4.

The convolution of filter and image is arrived at by computing the pairwise product of corresponding elements of the filter and the underlying portion of the image and summing them together. Notice that this is the same as computing the internal activation of the EES processing element shown in Figure 3. This means we can implement the EES neural net by using standard image processing hardware that supports convolution.

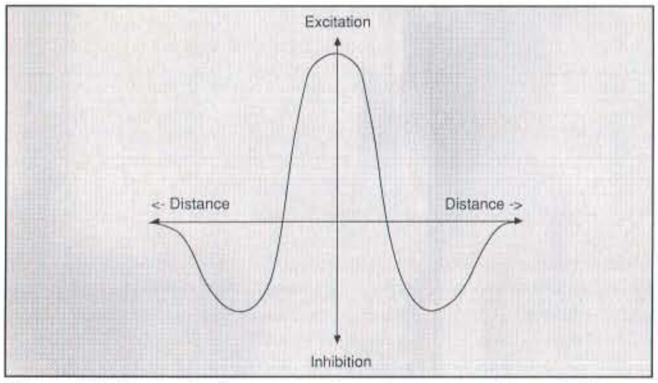


Figure 2: *Impact of a processing element on its neighbors. As the distance from the processing element in the center of the diagram increases, it excites its nearest neighbors. As the distance increases, the excitation turns to inhibition, and finally all effects cease. This curve is sometimes called a "Mexican Hat Function" or "Difference of Gaussians"*

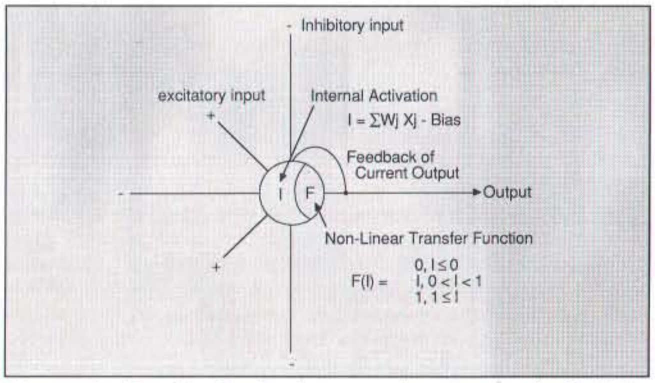

- Inhibitory input
excitatory input
+
Internal Activation
I = ΣWj Xj - Bias
Feedback of Current Output
-
I
F
Output
Non-Linear Transfer Function
F(I) = 0, I ≤ 0; I, 0 < I < 1; 1, 1 ≤ I
+
-

Figure 3: *Details of a single-processing element in the Edge Enhancement System. The "+" symbols indicate that the input will be excitatory. The "–" symbols indicate that the input will be inhibitory*

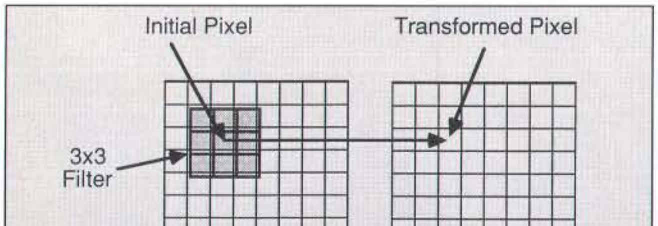


Figure 4: *Operation of a two-dimensional convolution. The 3 × 3 filter is laid over the initial image, centered at a particular pixel. The value of the matrix is multiplied by the pixel value just beneath it. The result of these pair-wise products are summed together to form the transformed pixel value in the transformed image*

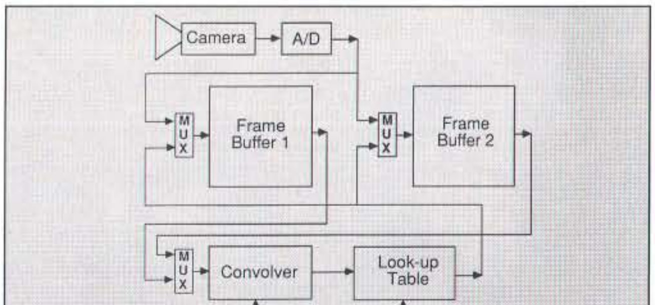


Figure 5: *Hardware block diagram of an image processing system adapted to be used to implement the Edge Enhancement System. The basic building blocks are a pair of frame buffers, a convolver, and a look-up table for performing nonlinear transformations*

Image filtering by use of convolutions is one of the cornerstones of machine vision. By properly selecting the coefficients of the filter, you can detect edges, create high- or low-pass filters, grow or shrink light regions, and quite a variety of other functions. You'll find more information on digital image filtering[2] at the end of this article. In practice, implementing an edge detector using a convolution filter is not difficult. The problem that arises is that of finding good filter coefficients, which do an effective job of finding the edges rather than losing or obscuring them.

A second commonly used technique in image processing is called a "look-up table." Just as the name implies, the value of a pixel is applied to the input of a look-up table (usually the address lines of a static RAM array) and a "transformed" value is produced at the output (the contents of that memory location). The mapping function is typically arbitrary and can be defined by the user.

Look-up tables are used to enhance contrast, convert images to black and white (from gray or color), and to produce special effects. The Cherry Coke commercials use this to make the can of Cherry Coke be in color and all else black and white. In our case, they can be used to implement a clamped linear transfer function. To implement a clamped linear transfer function in an 8-bit system, set the mapping RAM to output zero whenever an input in the range 0x80 through 0xff (negative values) is applied. For locations 0x00 through 0x7f, set the mapping RAM to output the same value as the input.

Implementing the EES

Both the convolution and look-up table techniques are such common tools that both are included in most commercial image processing systems. Together with a pair of frame buffers (also common), we can actually implement a very fast and moderately priced edge enhancement system. Companies that supply suitable hardware and software include Imaging Technologies (ITI), DataCube, Data Translation, and Matrox.

A block diagram of the hardware to implement the EES is shown in Figure 5. To set up the system, we load the block shown as "Filter Coefficients" with the coefficients from the MHF, and the look-up table "Transfer Function" with the values for a clamped linear transfer function; 7×7 is the minimum-sized convolution to use for the MHF. Some of the systems mentioned also support 9×9 and larger convolutions.

The sequence of processing is as follows:

1. Acquire an image from the camera to frame buffer 1.
2. Transform frame buffer 1 to frame buffer 2 using the MHF filter and clamped linear look-up table map function.
3. Transform frame buffer 2 to frame buffer 1 using the Mexican hat function filter and clamped linear look-up table map function.
4. Repeat steps 2 and 3 as many times as desired.

Because most systems are designed to work with small integers, it will be necessary to make the appropriate translations. This is an example of how neural-network technology can be grafted into existing technology to enhance its performance. With a little thought, it is possible to apply similar techniques to a variety of other problems.

Software Implementations of EES

When I began doing research on these filters for a project we are working on, I wanted something that would be easy to work with, and I could quickly try out a variety of parameters. After a little thought, I decided to try out my new

HARVARD SOFTWORKS

NUMBER ONE IN FORTH INNOVATION

(513) 748-0390 P.O. Box 69, Springboro, OH 45066

MEET THAT DEADLINE ! ! !

- Combine the raw power of extensible languages with the convenience of carefully implemented functions!
- Execute faster than optimized C!
- Compile 40,000 lines per minute!
- Stay totally interactive, even while compiling!
- Program at any level of abstraction from machine code thru application specific language with equal ease and efficiency!
- Modify routines without recompiling!
- Use source code for 2500 functions!
- Use data structures, control structures, and interface protocols from any other language!
- Implement that borrowed feature, often more efficiently than in its source!
- Use an architecture that supports very small programs or full megabyte ones with a single version!
- Forget chaotic syntax requirements!
- Use subroutine libraries written for other languages! More efficiently!
- Outproduce experienced programmers stuck using conventional languages!

HS/FORTH with FOOPS - The only full multiple inheritance object oriented language under MSDOS!

WAKE UP ! ! !

Forth is no longer a language that tempts programmers with "great expectations", then frustrates them with the need to reinvent simple tools expected in any commercial language.

HS/FORTH Meets Your Needs!

We don't shortchange you with promises. We provide implemented functions to help you complete your application quickly.

Don't judge Forth by public domain products or ones from vendors primarily interested in consulting - they profit from not providing needed tools! Public domain versions are cheap - if your time is worthless. Useful only in learning Forth's basics, they fail at showing its true potential. Not to mention being s-l-o-w.

You can't add extensibility to fossilized compilers. You can easily add features from any other language to **HS/FORTH**. Try our full modifiable multiple inheritance **OOPs** extension or our **Fortran** translator & mathpak!

HS/FORTH has more features and libraries than we can possibly describe here. **Please ask for our brochure!**

HS/FORTH uses MSDOS or PCDOS. Each level includes features of lower ones. Level upgrades: $25. plus price difference. Sources in ordinary text files.

STUDENT LEVEL **$145.**
text & scaled/clipped graphic windows, ellipses, splines, arcs, fills, turtles; powerful parsing, formatting, file and device I/O; shells; interrupt handlers

PERSONAL LEVEL **$245.**
software floating point, trig, transcendental, quad & scaled integer, automatic optimizer for assembly like performance.

PROFESSIONAL LEVEL **$395.**
hardware floating point, turnkey, seal, Rosetta Stone Dynamic Linker, round robin & time slice multitaskers, dynamic string manager, file blocks, sector mapped blocks, assemblers

PRODUCTION LEVEL **$495.**
Metacompiler: DOS/ROM/direct/indirect C data structures & struct compiler, TurboWindow-C graphics library,

Options:

FOOPS+ with multiple inheritance	**$ 75.**
286FORTH or **386FORTH**	**$295.**
TurboWindow-C (Metagraphics)	**$ 99.**
BTRIEVE for HS/FORTH (Novell)	**$199.**
ROMULUS HS/FORTH from ROM	**$ 95.**
FFORTRAN translator/mathpak	**$ 75.**

CIRCLE NO. 137 ON READER SERVICE CARD

copy of Lotus 1-2-3. The spreadsheet instance described in this section is the result of those efforts. Though I used Lotus 1-2-3, Release 3.0, it should be possible to implement this with most spreadsheet packages and computers that support a graphing option.

As it turns out, a variety of other techniques could have also been used to do this research. Listing One, page 114, shows a C program that implements the same functions as the spreadsheet, but without the nice graphics or ability to change data as easily as with the spreadsheet. Both the C language implementation and the spreadsheet implementation deal with the more limited problem of a one-dimensional data stream rather than the two-dimensional image processing we have been discussing. Later in this article, I'll discuss how to extend the one-dimensional model to two-dimensions.

Listing Two, page 114, shows the spreadsheet constructed. The numbers to the right are the row numbers. The letters along the bottom represent the column numbers. The Graph capability of Lotus 1-2-3 is used to display the results from processing the one-dimensional signal or data stream. Although not every aspect of the spreadsheet is discussed here, the entire spreadsheet is available on-line or on disk from *DDJ*.

The first step in constructing the spreadsheet is to set up the "static" data. This consists of all titles, the "Bias" (cell D7), "Low Pass Filter" (range C20. .C28), "MHF Filter" (range D20. .D28), and "Raw Input Data" (range E16. .E124). Everything else in the spreadsheet is computed. This static data is entered exactly as shown. For the Raw Input Data, 0.00 represents "black" and 1.00 represents "white." Intermediate values may be used. Be careful to put everything in the cell locations shown. After the spreadsheet is constructed, you can move things around to suit your taste.

The calculations for the Low Pass Output data are as follows, assuming that you have entered the static data in the rows and columns shown. Enter the following equation in cell B20:

```
+$C$20*E16+$C$21*E17
            +$C$22*E18+$C$23*E19
            +$C$24*E20+$C$25*E21
            +$C$26*E22+$C$27*E23
                       +$C$28*E24
```

or with Lotus 1-2-3, Release 3:

```
@SUMPRODUCT
        ($C$20. .$C$28,E16. .E24)
```

Then replicate cell B20 throughout the range B21. .B120. This column is labeled as "Graph A" as a reminder of which graph range to use to display it. (Line 14 of the spreadsheet.)

Calculations for the neural-network filter are done in a single step. Compute the internal activation and transfer function as follows:

```
@MAX(0.0,@MIN(1.0,
            $D$20*E16+$D$21*E17
           +$D$22*E18+$D$23*E19
           +$D$24*E20+$D$25*E21
           +$D$26*E22+$D$27*E23
              +$D$28*E24–$D$7))
```

or with Lottus 1-2-3, Release 3:

```
@MAX(0.0,@MIN(1.0,
         @SUMPRODUCT($D$20. .
          $D$28,E16. .E24)-$D$7))
```

Then replicate cell F20 throughout the range F21. .M120. The *@MAX(0, . .)* clamps the output so it can never go below zero. *@MIN(1, . .)* clamps the output so it can never go above one. The sum of the pair-wise products (or SUMPRODUCT) computes the effect of the neighborhood processing elements on the current one, and includes feedback of the current state. The *– $D$7* subtracts off the bias from the internal activation.

The first four and the last four cells in columns F through M are a copy of the values of the cells just prior to them. To replicate the values of the top of the columns, enter:

```
Cell F16: +F$20
```

Then replicate it throughout the range F16. .M19. To replicate the values at the bottom of the columns, enter:

```
Cell F121: +F$120
```

Then replicate it throughout the range F121. .M124. The computation portion of the spreadsheet is now complete. Use the graphing feature of your spreadsheet to construct the graphs described in Figure 6. These two graphs will be used to display the processing effects of various types of inputs and filters on the output data.

Testing the Spreadsheet Implementation

Having constructed the spreadsheet just described, the graph EES should look like the one in Figure 7a. Figure 7b is the same graph with the input range (Range B) reset, so it shows only the output of the network as it evolves. Figure 7c shows the input data and the final (eighth) iteration of the network with intermediate ranges reset (Ranges C, D, E).

The edge data for this experiment was selected to show profiles of two kinds of edges often found in images. In the first kind, light shines on a curved edge or rounded edge resulting in a gradation in intensities. The gradually changing light intensities on the left side of the graph are typical of this kind of edge. The second kind of edge is a ragged edge such as from torn metal. This type shows wide variations in gray level due to specular reflectivity as well as sharp variations in the curvature of the material. This is shown as the very noisy edge on the right of center of the diagram. Notice that the EES does a very nice job of sharpening both edges.

To the far right is a small "blip" in intensities. This blip is of the same magnitude as the one in the center of the main pulse. Notice that the EES was able to pick this out, because of its contrast to the background, while ignoring the noise on the top of the pulse. A little experimentation will show that this is quite a powerful technique. The bias value (in cell D7) can be changed to alter the sensitivity to various features. Changing the shape of the MHF also changes the nature of edges detected. Figure 8 shows the MHF used in the filter.

As it turns out, both edges used in this test tend to be difficult to find using standard image processing techniques. Figure 9 shows what happens when a simple sobel operator is applied to the input. The resulting derivative function does not provide much information about where the edges might be. The problem is not that such a filter is difficult to implement, but that finding a set of coefficients and a filter length, which enhances the edges rather than missing or obscuring them, is a highly heuristic and often frustrating task. My own experience is that it is sometimes impossible.

There are a variety of experiments you can do with the edge enhancement system. One of the things you

EES (Edge Enhancement System):

Format: Lines only

Graph	Range	Contents
B	E16. .E124	input data
C	F16. .F124	1st iteration
D	H16. .H124	3nd iteration
E	J16. .J124	5th iteration
F	M16. .M124	8th iteration

HIGHPASS (High Pass Filter):

Format: Lines only

Graph	Range	Contents
A	B20. .B120	Low-pass filtered data
B	E20. .E120	input data

Figure 6: *Constructing the graphs*

will discover is that the system can be sensitive to the shape of the MHF as well as the bias. In some ranges, the detected edges actually set up standing waves, which emanate out from the edges!

How It Works

As I mentioned at the beginning of this article, the EES is an engineering approach to image or signal processing based on biological insights. It was developed heuristically starting with a biological model and studying it until the mechanics of its operation were well understood. As such, there is no formal theory of operation.

Functionally what happens is that the MHF acts as a difference of Gaussians filter. The transfer function clips the negative part of the output, leaving only the positive center peak when the

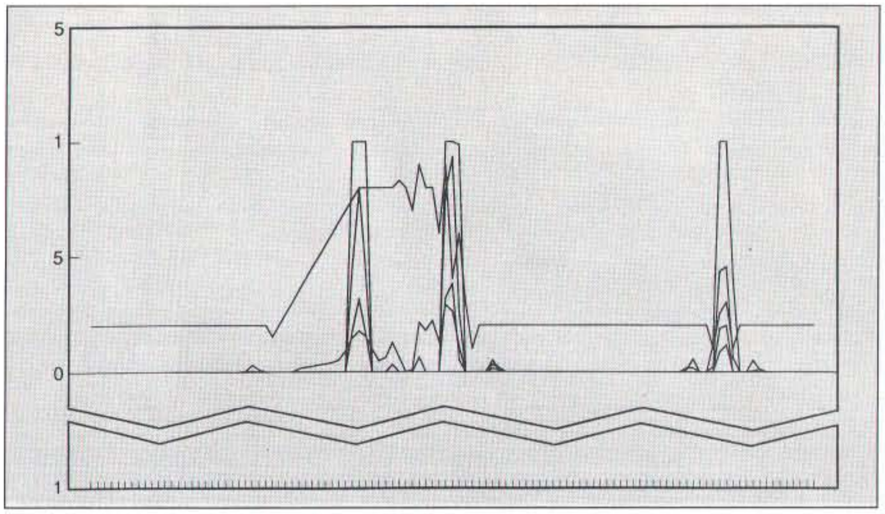


Figure 7a: *Output of graph EES of the Edge Enhancement System spreadsheet. The inputs, as well as the network outputs after 1, 3, 5, and 8 iterations, are shown super-imposed*

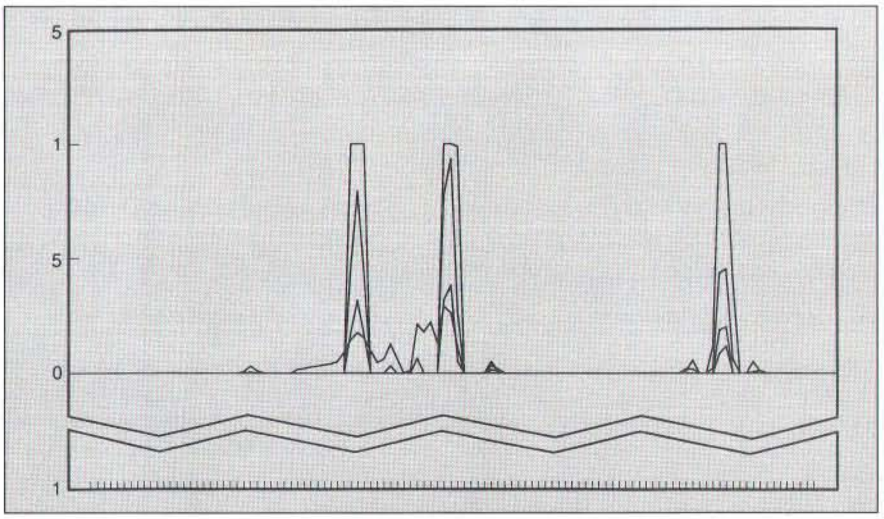


Figure 7b: *Output of the network as it evolves. This shows only the network outputs after 1, 3, 5, and 8 iterations*

```
 0.0 -0.1 -0.1 -0.2 -0.2 -0.2 -0.1 -0.1  0.0
-0.1 -0.5 -0.6 -0.6 -0.6 -0.6 -0.6 -0.5 -0.1
-0.1 -0.6 -0.4 -0.2 -0.2 -0.2 -0.4 -0.6 -0.1
-0.2 -0.6 -0.2  1.0  1.0  1.0 -0.2 -0.6 -0.2
-0.2 -0.6 -0.2  1.0  1.8  1.0 -0.2 -0.6 -0.2
-0.2 -0.6 -0.2  1.0  1.0  1.0 -0.2 -0.6 -0.2
-0.1 -0.6 -0.4 -0.2 -0.2 -0.2 -0.4 -0.6 -0.1
-0.1 -0.5 -0.6 -0.6 -0.6 -0.6 -0.6 -0.5 -0.1
 0.0 -0.1 -0.1 -0.2 -0.2 -0.2 -0.1 -0.1  0.0
```

Example 1: *A two-dimensional version of the EES*

The Berkeley Utilities

UNIX COMMANDS FOR MS-DOS

Now you can tap the power of UNIX on your DOS machine . . .

- 32 UNIX commands with all System V.3 and BSD 4.3 options plus a number of DOS-specific enhancements
- Includes AWK, GREP, FIND, MAKE, DIFF, ED, SED, and SORT
- Each utility is standalone, uses video attributes, and has a built-in Help screen
- Comprehensive manuals, including an introduction to UNIX for DOS users

The Best of Both Worlds

Requires 256K and DOS 3.0 or higher. A hard disk is recommended.

Order Now. $125.

VISA or MasterCard accepted.

718-638-2240

30 DAY MONEY-BACK GUARANTEE

Also from OPENetwork . . .

Mi-Shell

The programmable DOS Shell that requires less than 8K RAM

DELTA

The ideal comparison tool for programmers

TextLib

Our screen & keyboard management library for C developers

Write or call our BBS for information

OPENetwork

POWER TOOLS FOR POWER USERS

215 Berkeley Place, Brooklyn, New York 11217
Voice: 718·638·2240 BBS: 718·638·2239

CIRCLE NO. 163 ON READER SERVICE CARD

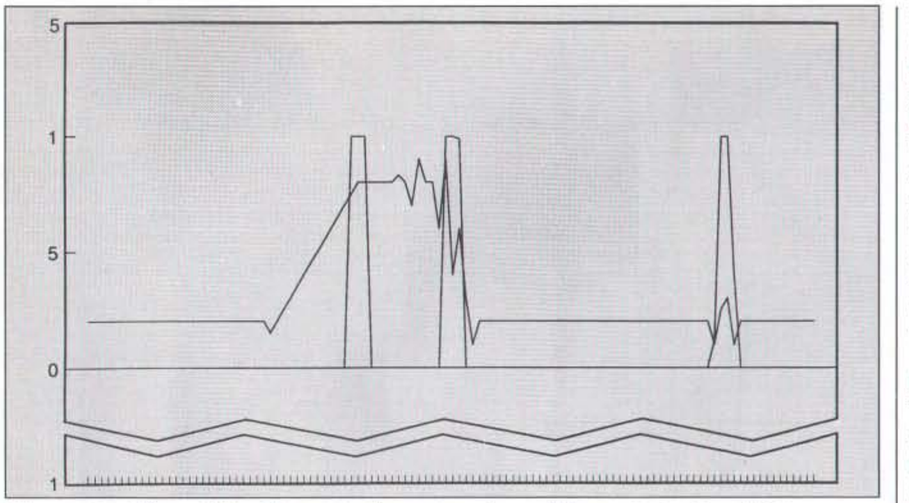


Figure 7c: *The original input to the network, and the output after eight iterations are shown super-imposed*

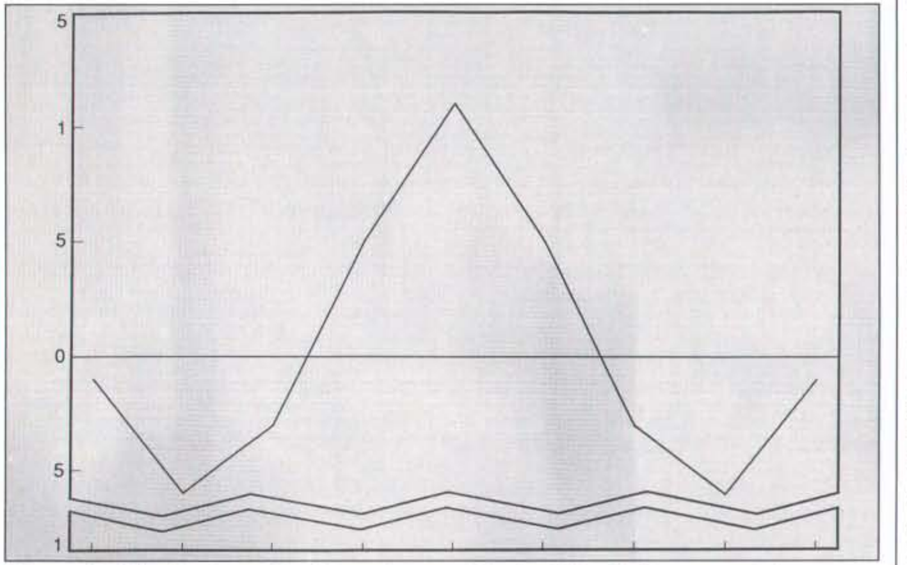


Figure 8: *The Mexican Hat Function used in the Edge Enhancement System*

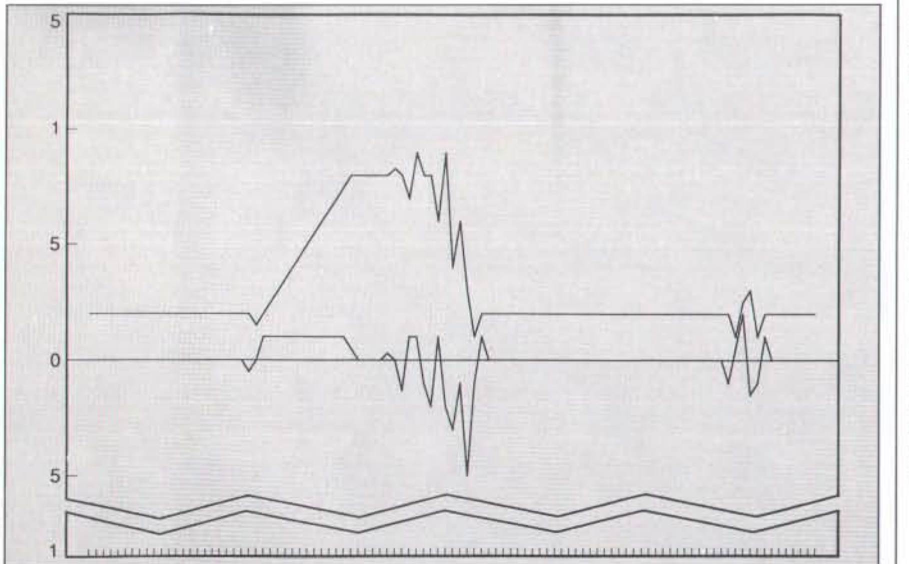


Figure 9: *Results of applying a sobel (simple edge detection) filter to the input data. The sobel computes the derivative of the image*

filter is directly over the edge. When the MHF is applied again to the resulting output, it will tend to enhance single peaks but reduce plateaus. As such, lots of noise in the vicinity of an edge will be ignored. However, a single substantial variation against a constant background will be significantly enhanced. Iterating on this process eventually results in groups of saturated processing elements, at most the width of the excitatory part of the MHF. All other processing elements are turned off.

Extending the EES to Two-Dimensions

The same principles used in developing the one-dimensional EES apply to the two-dimensional version. Instead of using a single-dimensional vector, a two-dimensional matrix is used. One example of a 9 × 9 MHF is shown in Example 1.

The process of computing the convolution (sum of pair-wise products) with the corresponding portion of a pixel array is the same. Likewise, the clamped linear transfer function uses the same equation used in the spreadsheet. Implementing this on an image processing system will require converting everything to work with small integers, but the process is quite straightforward.

Summary

Insights from the operation of the human eye can be used to build improved image enhancement systems, particularly edge enhancement systems. The basic mechanisms involved are capable of turning an image (or one-dimensional signal) with fuzzy and noisy edges into a sharp clean edge-enhanced image.

This technology can enhance the solution of a variety of problems including character recognition, part tracking, part inspection, printed circuit board inspection, ultrasonic image interpretation, target recognition, and so on. Enough similarities exist with traditional image processing techniques that these neural networks can be implemented with traditional image processing hardware and software systems.

References

1. Carver, Mead. *Analog VLSI and Neural Systems*, Addison-Wesley, Reading, Mass.: 1989.

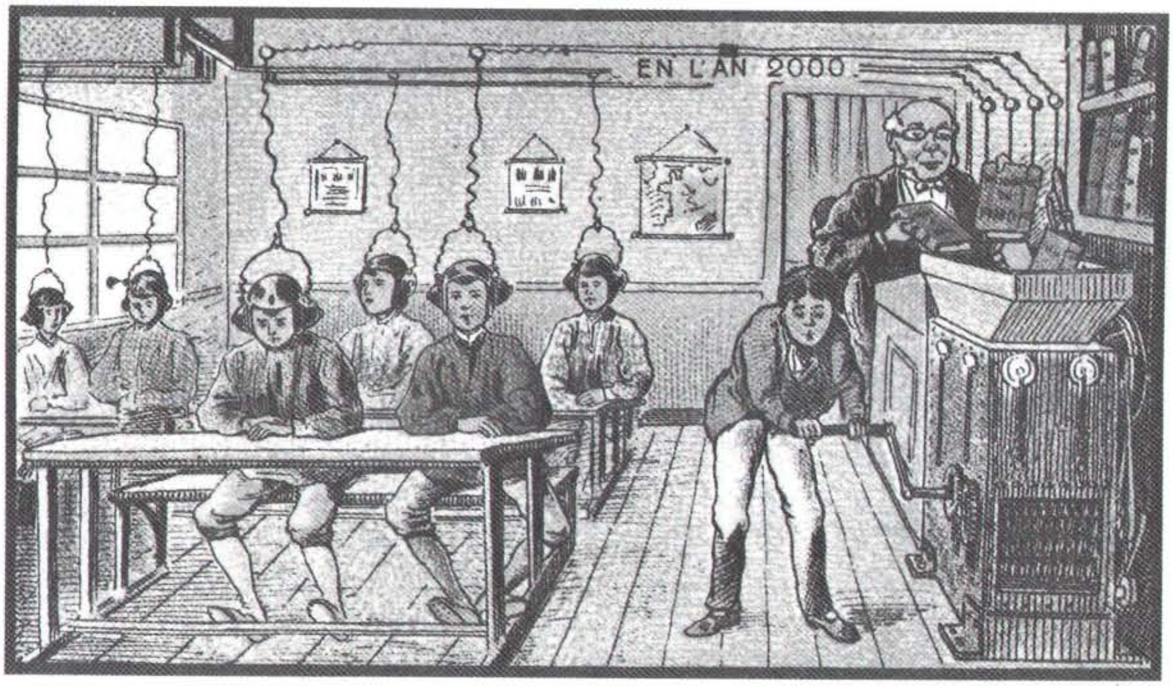
2. Tzay Y. Young, King-Sun Fu. *Handbook of Pattern Recognition and Image Processing*, Academic Press, Orlando, Fla.: 1986.

DDJ

(Listings begin on page 114.)

Vote for your favorite feature/article.
Circle Reader Service **No. 8.**

Microsoft creates software giants in laboratory.



A class at Microsoft® University will go straight to your head.

Reason being, your course instructors work for us, Microsoft. The country's leading developer of software.

Better still, they do their teaching in a laboratory setting that gives you two big advantages: Hands-on experience. And software you've developed that's yours to keep.

It's the fastest way to learn the latest technology being utilized in today's popular programs. Giving you, and your corporation, a big jump in developing software.

Courses are offered in several powerful systems platforms, including Microsoft OS/2, Microsoft OS/2 Presentation Manager and Windows.™ And innovative networking technologies like LAN Manager and Microsoft SQL Server.

To get more information and a free copy of the Microsoft University catalog, call (800) 541-1261, Dept. K89.

As a graduate, you'll soon be writing better applications, faster.

Making your career, and your company, grow by leaps and bounds.

Microsoft *University*

Listing One *(Text begins on page 16.)*

```
//////////////////////////////////////////////////////////////
// BAM.HPP  Provide vector, matrix, vector pair, matrix, BAM matrix, and
// BAM system classes and methods to implement BAM system concept.
// Extended note:
// This is an implementation of the concept of Bidirectional
// Associative Memories as developed by Bart Kosko and others.
// It includes the extended concept introduced by Patrick Simpson
// of the "BAM System". Where reasonable Simpson's notation has been
// been maintained. The presentation benefits from C++ and OOP, in that
// (I believe) it is both easier to understand than a "pseudocode" version,
// yet more precise (in that it works!)
// Developed with Zortech C++ Version 2.0 -- Copyright (c) Adam Blum, 1989,90

#include<stdlib.h>
#include<io.h>
#include<stdio.h>
#include<string.h>
#include<limits.h>
#include<ctype.h>
#include<stream.hpp>
#define D(x) {if (trace){x}}
extern int trace;

// where are Zortech's min,max?
#define max(a,b)        (((a) > (b)) ? (a) : (b))
#define min(a,b)        (((a) < (b)) ? (a) : (b))

// will be changed to much higher than these values
const ROWS=16;  // number of rows (length of first pattern)
const COLS=8;   // number of columns (length of second pattern)
const MAXMATS=10; // maximum number of matrices in BAM system
const MAXVEC=16; // default size of vectors

class matrix;
class bam_matrix;
class vec {
        friend class matrix;
        friend class bam_matrix;
        friend class bam_system;
                int n;
                int *v;
        public:
                // see BAM.CPP for implementations of these
                vec(int size=MAXVEC,int val=0); // constructor
                ~vec(); // destructor
                vec(vec &v1); // copy-initializer
                int length();
                vec& operator=(const vec& v1); // vector assignment
                vec& operator+(const vec& v1); // vector addition
                vec& operator+=(const vec& v1); // vector additive-assignment
                vec& operator*=(int i); // vector multiply by constant
                // supplied for completeness, but we don't use this now
                int operator*(const vec& v1); // dot product
                vec operator*(int c); // multiply by constant
                // vector transpose multiply needs access to v array
                int operator==(const vec& v1);
                int& operator[](int x);
                friend istream& operator>>(istream& s,vec& v);
                friend ostream& operator<<(ostream& s, vec& v);
}; //vector class

class vecpair;

class matrix {
        protected:
        // bam_matrix (a derived class) will need to use these members
        // preferred to "friend class", since there may be many derived
        // classes which need to use this
                int **m; // the matrix representation
                int r,c; // number of rows and columns
        public:
                // constructors
                matrix(int n=ROWS,int p=COLS);
                matrix(const vec& v1,const vec& v2);
                matrix(const vecpair& vp);
                matrix(matrix& m1); // copy-initializer
                ~matrix();
                int depth();
                int width();
                matrix& operator=(const matrix& m1);
                matrix& operator+(const matrix& m1);
                matrix& operator+=(const matrix& m1);
                vec colslice(int col);
                vec rowslice(int row);
                friend ostream& operator<<(ostream& s,matrix& m1);
}; // matrix class

class vecpair {
        friend class matrix;
        friend class bam_matrix;
        friend class bam_system;
                int flag; // flag signalling whether encoding succeeded
                vec a;
                vec b;
        public:
                vecpair(int n=ROWS,int p=COLS); // constructor
                vecpair(const vec& A,const vec& B);
                vecpair(const vecpair& AB); // copy initializer
                ~vecpair();
                vecpair& operator=(const vecpair& v1);
                int operator==(const vecpair& v1);
                friend istream& operator>>(istream& s,vecpair& v);
                friend ostream& operator<<(ostream& s,vecpair& v);
                friend matrix::matrix(const vecpair& vp);
};

class bam_matrix: public matrix {
        private:
                int K; // number of patterns stored in matrix
                vecpair *C; // actual pattern pairs stored
                int feedthru(const vec&A,vec& B);
                int sigmoid(int n); // sigmoid threshold function
        public:
                bam_matrix(int n=ROWS,int p=COLS);
                ~bam_matrix();
                // but we supply it with the actual matrix A|B (W is implied)
                void encode(const vecpair& AB); // self-ref version
                // uncode only necessary for BAM-system
                void uncode(const vecpair& AB); // self-ref version
                vecpair recall(const vec& A);
                int check();
                int check(const vecpair& AB);
                // Lyapunov energy function: E=-AWBtranspose
                int energy(const matrix& m1); // Lyapunov energy function
}; // BAM matrix

class bam_system {
                bam_matrix *W[MAXMATS];
                int M; // number of matrices
        public:
                bam_system(int M=1);
                ~bam_system();
                void encode(const vecpair& AB);
                vecpair& recall(const vec& A);
                // train equiv. to Simpson's encode of all pairs
                void train(char *patternfile);
                friend ostream& operator<<(ostream& s,bam_system& b);
}; // BAM system class
```

End Listing One

Listing Two

```
////////////////////////////////////////
// BAM.CPP Provide vector, matrix, vector pair, matrix, BAM matrix, and BAM
// system classes to implement BAM systems
// Extended note:
// This is an implementation of the concept of Bidirectional
// Associative Memories as developed by Bart Kosko and others.
// It includes the extended concept introduced by Patrick Simpson
// of the "BAM System". Where reasonable Simpson's notation has been
// been maintained. The presentation benefits from C++ and OOP, in that
// (I believe) it is both easier to understand than a "pseudocode" version,
// yet more precise (in that it works!)
// Developed with Zortech C++ Version 2.0 -- Copyright (c) 1989,90 Adam Blum

#include"bam.hpp"

/////////////////////////////////////
// vector class member functions

vec::vec(int size,int val) {
        v = new int[size];
        n=size;
        for(int i=0;i<n;i++)
                v[i]=0;
} // constructor
vec::~vec() { delete v;} // destructor
vec::vec(vec& v1) // copy-initializer
{
        v=new int[n=v1.n];
        for(int i=0;i<n;i++)
                v[i]=v1.v[i];
}
vec& vec::operator=(const vec& v1)
{
        delete v;
        v=new int[n=v1.n];
        for(int i=0;i<n;i++)
                v[i]=v1.v[i];
        return *this;
}
vec& vec::operator+(const vec& v1)
{
        vec sum(v1.n);
        sum.n=v1.n;
        for(int i=0;i<v1.n;i++)
                sum.v[i]=v1.v[i]+v[i];
        return sum;
}
vec& vec::operator+=(const vec& v1)
{
        for(int i=0;i<v1.n;i++)
                v[i]+=v1.v[i];
        return *this;
}
vec vec::operator*(int c)
{
        vec prod(length());
        for(int i=0;i<prod.n;i++)
                prod.v[i]=v[i]*c;
        return prod;
}
int vec::operator*(const vec& v1) // dot-product
{
        int sum=0;
        for(int i=0;i<min(n,v1.n);i++)
                sum+=(v1.v[i]*v[i]);
        //D(cout << "dot product " << *this << v1 << sum << "\n";)
        return sum;
}
int vec::operator==(const vec& v1)
{
        if(v1.n!=n)return 0;
        for(int i=0;i<min(n,v1.n);i++){
                if(v1.v[i]!=v[i]){
                        return 0;
                }
        }
        return 1;
```

```
}
int& vec::operator[](int x)
{
        if(x<length() && x>=0)
                return v[x];
        else
                cout << "vec index out of range";
}
int vec::length(){return n;} // length method

istream& operator>>(istream& s,vec &v)
// format: list of ints followed by ','
{
        char c;
        v.n=0;
        v.v=new int[MAXVEC];
        for(;;){
                s>>c;
                if(s.eof())return s;
                if(c==',')return s;
                if(isspace(c))continue;
                v.v[v.n++]=((c!='0')?1:-1);
        }
}
ostream& operator<<(ostream& s, vec& v)
// format: list of ints followed by ','
{
        for(int i=0;i<v.n;i++)
                s << (v.v[i]<0?0:1);
        s << ",";
        return s;
}

////////////////////////////////
// matrix  member functions
matrix::matrix(int n,int p)
{
        //D(cout << "Constructing " << n << " x " << p << " matrix.\n";)
        m=new int *[n];
        for(int i=0;i<n;i++){
                m[i]=new int[p];
                for(int j=0;j<p;j++)
                        m[i][j]=0;
        }
        r=n;
        c=p;
} // constructor
matrix::matrix(const vecpair& vp)
{
        //D(cout << "Constructing matrix from: " << vp;)
        r=vp.a.length();
        c=vp.b.length();
        m=new int *[r];
        for(int i=0;i<r;i++){
                m[i]=new int[c];
                for(int j=0;j<c;j++)
                        m[i][j]=vp.a.v[i]*vp.b.v[j];
        }
}// constructor
matrix::matrix(const vec& v1,const vec& v2)
{
        //D(cout << "Constructing matrix from " << v1 << v2 << "\n";)
        r=v1.length();
        c=v2.length();
        m=new int *[r];
        for(int i=0;i<r;i++){
                m[i]=new int[c];
                for(int j=0;j<c;j++)
                        m[i][j]=v1.v[i]*v2.v[j];
        }
}// constructor
matrix::matrix(matrix& m1) // copy-initializer
{
        //D(cout << "matrix copy-initializer\n"; )
        r=m1.r;
        c=m1.c;
        m=new int *[r];
        for(int i=0;i<r;i++){
                m[i]=new int[c];
                for(int j=0;j<c;j++)
                        m[i][j]=m1.m[i][j];
        }
}
matrix::~matrix()
{
        for(int i=0;i<r;i++)
                delete m[i];
        delete m;
} // destructor
matrix& matrix::operator=(const matrix& m1)
{
        for(int i=0;i<r;i++)
                delete m[i];
        r=m1.r;
        c=m1.c;
        m=new int*[r];
        for(i=0;i<r;i++){
                m[i]=new int[c];
                for(int j=0;j<r;j++)
                        m[i][j]=m1.m[i][j];
        }
        return *this;
}
matrix& matrix::operator+(const matrix& m1)
{
        matrix sum(r,c);
        for(int i=0;i<r;i++)
                for(int j=0;j<r;j++)
                        sum.m[i][j]=m1.m[i][j]+m[i][j];
        return sum;
```

(Listing continued on page 86)

Windows® Text Editing for the serious programmer

SPEED Edit

SpeedEdit, the only program development environment/text editor for use with MS-Windows, Presentation Manager®, DOS, OS/2 and Unix®. Call or write:

Bradford Business Systems, Inc.
25301 Cabot Road, Suite 201, Laguna Hills, California 92653
(714) 859-4428

Windows is a registered trademark of Microsoft Corp., Presentation Manager is a trademark of IBM Corp., Unix is a trademark of ATT.

CIRCLE NO. 406 ON READER SERVICE CARD

MICROSOFT, TURBO AND MIX POWER C PROGRAMMERS...C WINDOWS TOOLKIT™ PUTS YOU IN CHARGE OF VIDEO *and* DATA-ENTRY

NEW FEATURES

- *Comprehensive data-entry and verification procedures with scanf( ) type syntax.*
- *Predefined data-entry functions for dates, SSNs, telephone numbers, currency amounts with commas, time, scientific notation, etc.*
- *Turn any window into an editor with your choice of keyboard mapping (Wordstar™ by default).*
- *Virtual windows up to available memory.*
- *Save windows to disk.*
- *Nested windows.*
- *Create and edit VGA/MCGA/EGA/Hercules fonts of any size.*

NEW VERSION 2.0

WINDOWING FUNCTIONS

- Create pop-up windows
- Create pull-down menus
- Create spreadsheet menus
- Create context-sensitive help screens
- Store windows in RAM or on disk
- Modify windows out of view
- Use 8 different types of exploding windows
- Scroll and move windows

SYSTEM CONTROL

- Device-independent output
- Supports VGA/MCGA/EGA/Hercules/ CGA/MDA
- Detect the types of video adapter installed
- Switch between adapters
- Detect ANSI.SYS
- Detect the enhanced keyboard
- Disable the video signal
- Delay program execution to microsecond resolution

Full source code included — No run-time royalties.

VGA/EGA SUPPORT

- Use all 64 VGA/EGA colors
- Use VGA 29/43/50 line modes
- Use EGA 43-line mode
- Use 2 fonts simultaneously
- Design and edit custom fonts with FONTEDIT ™our font editor (included)
- Smooth scroll and pan the screen

FAST SCREEN I/O

- Write formatted output (like **printf( )**)
- Get snow-free output on the CGA
- Use direct screen writes or the BIOS

HERCULES SUPPORT

- Create and edit your own Ramfonts™
- Detect the presence of a Hercules Card
- Detect Ramfont™ support
- Load and store Ramfonts™ to RAM or disk
- Switch between modes

Includes 230 page manual — 30-day Money-Back Guarantee

Magna Carta SOFTWARE

P.O. Box 475594
Garland, TX 75047-5594

Requires: IBM PC, XT,AT, PS/2 or compatible.
Supports Microsoft C/Quick C, Borland Turbo C, Mix Power C

CALL (214) 226-6909

Only $99.95

(Texas Residents add 8% sales tax.)

CIRCLE NO. 148 ON READER SERVICE CARD

Listing Two *(Listing continued, text begins on page 16.)*

```
}
matrix& matrix::operator+=(const matrix& m1)
{
        //D(cout << "matrix additive assignment\n";)
        for(int i=0;i<r&&i<m1.r;i++)
                for(int j=0;j<c&&j<m1.c;j++)
                        m[i][j]+=(m1.m[i][j]);
        return *this;
}
vec matrix::colslice(int col)
{
        vec temp(r);
        for(int i=0;i<r;i++)
                temp.v[i]=m[i][col];
        return temp;
}
vec matrix::rowslice(int row)
{
        vec temp(c);
        for(int i=0;i<c;i++)
                temp.v[i]=m[row][i];
        return temp;
}
int matrix::depth(){return r;}
int matrix::width(){return c;}

ostream& operator<<(ostream& s,matrix& m1)
// print a matrix
{
        for(int i=0;i<m1.r;i++){
                for(int j=0;j<m1.c;j++)
                        s << m1.m[i][j] << " ";
                s << "\n";
        }
}
//////////////////////////////////////////
// vecpair  member functions
// constructor
vecpair::vecpair(int n,int p) { }
vecpair::vecpair(const vec& A,const vec& B) {a=A;b=B;}
vecpair::vecpair(const vecpair& AB) {*this=vecpair(AB.a,AB.b);}
vecpair::~vecpair() {} // destructor
vecpair& vecpair::operator=(const vecpair& v1)
{
        a=v1.a;
        b=v1.b;
        return *this;
}
int vecpair::operator==(const vecpair& v1)
{
        return  (a == v1.a) && (b == v1.b);
}
istream& operator>>(istream& s,vecpair& v1)
// input a vector pair
{
        s>>v1.a>>v1.b;
        return s;
}
ostream& operator<<(ostream& s,vecpair& v1)
// print a vector pair
{
        return s<<v1.a<<v1.b<<"\n";
}
////////////////////////////////////
//bam_matrix  member functions
bam_matrix::bam_matrix(int n,int p):(n,p)
{
        // the maximum number of pattern pairs storable
        // is around min(n,p) where n and p are
        // the dimensionality of the matrix
        C=new vecpair[min(n,p)*2];
        K=0;
}

bam_matrix::~bam_matrix()
{
} // destructor
void bam_matrix::encode(const vecpair& AB)
// encode a pattern pair
{
        //D(cout << "BAM Matrix encoding: " << AB;)
        matrix T(AB);
        (*this)+=T; // add the matrix transpose to the current matrix
        C[K]=AB;
        K++;
}
void bam_matrix::uncode(const vecpair& AB)
// get rid of a stored pattern (by encoding A-B complement)
{
        //D(cout << "uncode\n";)
        vec v=AB.b*-1;
        matrix T(AB.a,v); // T is A transpose B complement
        *this+=T;// add the matrix transpose to the current matrix
        K--;
}
vecpair bam_matrix::recall(const vec& A)
// BAM Matrix recall algorithm (used by BAM SYSTEM recall)
{
        int givenrow=(A.length()==width());
        D(cout<<"BAM matrix recall of" << A << givenrow?"(row)\n":"(col)\n";)
        vec B(givenrow?depth():width(),1);
        for(;;){ // feed vectors through matrix until "resonant" pattern-pair
                feedthru(A,B);
                if(feedthru(B,A))break; // stop when returned A = input A
        }
        D(cout<< "resonant pair " << A << "\n and " << B << "\n";)
        if(givenrow)
                return vecpair(B,A);
        else
                return vecpair(A,B);
}
int bam_matrix::feedthru(const vec&A,vec& B)
{
        //D(cout << "Feeding " << A << "\n"; )
        vec temp=B;int n;
        for(int i=0;i<B.length();i++){
                if(A.length()==width())
                        n=sigmoid(A*rowslice(i));
                else
                        n=sigmoid(A*colslice(i));
                if(n)
                        B.v[i]=n;
        }
        return B==temp;
}
int bam_matrix::sigmoid(int n)
// VERY simple (but classic one for BAM) threshold function
//
//            1 --------------
//              :
//  - -----------          +
//           -1
{
        if(n<0)return -1;
        if(n>0)return 1;
        return 0;
}
int bam_matrix::check()
// check to see if we have successfully encoded pattern-pair into this matrix
{
        D(cout << "Check BAM matrix for " << K << " pattern pairs\n";)
        vecpair AB;
        for(int i=0;i<K;i++){
                AB=recall(C[i].a);
                if(!(AB==C[i])){
                        D(cout <<"failed check\n ";)
                        return 0;
                }
        }
        D(cout << "passed check\n ";)
        return 1;
}
int bam_matrix::check(const vecpair& AB)
{
        // different check routine for orthogonal construction BAM
        //check to see energy of present pattern pair to matrix
        // is equal to orthogonal BAM energy
        matrix T(AB);
        return energy(T)== -depth()*width();
}
int bam_matrix::energy(const matrix& m1)
{
        int sum=0;
        for(int i=0;i<depth();i++)
                for(int j=0;j<width();j++)
                        sum+=(m1.m[i][j]*this->m[i][j]);
        D(cout << "Energy of matrix " << -sum << "\n";)
        return -sum;
}

//////////////////////////////////////////
// bam system  functions
// top level of system (for now)

// constructor
bam_system::bam_system(int n)

{
        M=n;
        for(int i=0;i<M;i++)
                W[i]=new bam_matrix;
}
bam_system::~bam_system() // destructor
{
        for(int i=0;i<M;i++)
                delete W[i];
}
void bam_system::encode(const vecpair& AB)
// encode the pattern pair AB into the BAM system
{
        D(cout << "BAM System encode\n";)
        for(int h=0;h<M;h++){
                W[h]->encode(AB);
                if(!W[h]->check())
                        W[h]->uncode(AB);
                else
                        break;
        }
        if(h==M){ // all matrices full, add another
                if(h<MAXMATS){
                        W[M]=new bam_matrix();
                        W[M]->encode(AB);
                        M++;
                }
                else{
                        cout << "BAM System full\n";
                        exit(1);
                }
        }
}
vecpair& bam_system::recall(const vec& A)
// presented with pattern A, recall will return pattern-PAIR
{
        vecpair XY[MAXMATS];matrix *M1,*M2;
        int E,minimum=0,emin=INT_MAX;
        D(cout << "BAM System recall\n";)
        for(int h=0;h<M;h++){
                XY[h]=W[h]->recall(A);
                D(cout << h <<"-th matrix, returned vecpair "<< XY[h];)
                M1=new matrix(XY[h]);
```

(Listing continued on page 88)

Language

AI-Expert Systems

ARITY Combination Pkg	974
Exsys Professional	694
PC Scheme LISP-by TI	74
Personal Consultant Plus	2482
Turbo Prolog Toolbox	70
TURBO PROLOG V2.0	110
VP Expert	219

Assemblers

Advantage Disassembler	279
Incra	159
MS Macro Asm	97
Turbo Assembler/Debugger	98
Visible Computer 80286	84

Basic & Addons

DB/Lib	129
Exim Toolkit	80
MS Basic 6.0	199
MS QuickBASIC V4.5	69
QBase	124
QuickPak Professional	129
Softcode	69
True Basic	69
Turbo Basic	69

C Language-Compilers

AZTEC C86-Commercial	699
High C-286-by MetaWare	554
Instant C/16M	715
Lattice C-V6	199
Microsoft C 5.1-w/CodeView	295
Microsoft QuickC w/QASM	135
Turbo C-by Borland	105
Turbo C Professional	175
Watcom C7.0-highly optimized	315

Cobol

MS Cobol V3.0	629
Microfocus COBOL w/tools	1,345
Realia COBOL	844

Fortran

FOR_C-FORTRAN '77 to C	670
I/O Pro-screens	124
Lahey FORTRAN F77L	525
Lahey FORTRAN F77L-EM/16	785
Lahey Personal FORTRAN	84
MS FORTRAN-CodeView	295
RM Fortran	494
WatFor	349

Modula-2

Logitech Dev. Sys.	195
Topspeed 3 pak	159

Object-Oriented Programming

ACTOR	423
Language Extension 1	95
C_talk	137
C_talk/Windows	375
Smalltalk/V	85
EGA/VGA Color Ext.	45
Goodies #1, #2 or #3	45
Smalltalk/V 286	169
Zortech C++	155
w/source	205
Zortech C++ Tools V2.0	135

Other Languages

IntegrAda	724
Janus Ada	369
Meridian Ada	1,119
MuLISP-87	219
PC/Forth +	199
Personal Rexx	129
RPG II complete	1290

Turbo Pascal

graphics-Menu	134
Turbo ASYNCH PLUS	125
Turbo Pascal 5.0	105
Turbo Pascal 5.0 Professional	175
Turbo POWER TOOLS PLUS	124
Turbo Programmer	444

DBMS

Database Development

DBman	285
DBXL	199
Clarion Prof.	459
Clipper	399
dBASE III Plus	450
Force	585
FoxBASE +	239
FoxBASE + /386	395
Fox Pro Lan	745
Fox Pro SU	579
Magic PC	249
Paradox 3.0	469
QuickSilver	379
R:Base for DOS	559

dBASE Library

Artful	189
CodeBASE IV	215
Funky	175
IDL	125
Netlib	185
Proclip V3.5	119

File Management

Btrieve	180
CBTREE	169
C-Index for MSC	175
C-Index Plus	329
C-ISAM	325
CQL	199
w/PASS	349
c-tree	299
d-tree	395
r-tree	239
dBC III PLUS	435
db_FILE RETRIEVE	265
Essential B-Tree w/source	149
FABS Plus	172
Netware SQL	489
pBase	185
Turbo Programmer/C	449
XDB-C	555
XQL	599
Xtrieve PLUS	459

Addons

C Communications

New! C Async Lib. by Silverware	209
BreakOut II	99
C ASYNCH MANAGER	145
Essential Communications	159
Greenleaf Comm Library	219
Greenleaf ViewComm	450
Lattice Comm. Library	179

Dbase Addons

AdComm 99	105
Alink	179
CLEAR + for dBASE	139
Dbase Online V2.0	125
dBug-source debugger	170
dBX-dBASE III to C	460
dClip	189
dGE-business graphics	174
Documentor	225
dQuery	165
dSalvage	79
FLIPPER Graphics Library	175
Friendly Finder	90
Genifer-code generator	259
R&R Code Generator	125
R&R Relational Reportwriter	129
Scrimmage-screen/menu	129
SilverComm Library V2	149
Silver Paint Library	79
Steve Straly Tools	169
Sycero db-single user	444
Tom Rettig's Library	78
UI Programmer-Dev's Version 2.0	419

General Addons

C Tools Plus-V5.0	105
C Utilities-by Essential	154
Greenleaf Functions	159
Greenleaf SuperFunctions	209

Graphic Addons

Essential Graphics	229
Graphic C	329
Greenleaf Makeform-DOS	90
GSS Dev't Toolkit	505
Halo '88-140 + devices	275
Hoops 3D Graphics	494
MetaWINDOW/PLUS	269
PCX Programmer's Toolkit	175
w/source	435
Pizazz Plus	1211
Turbo Meta-Menu w/source	204
Turbo Plus-Nostradamus	114

Text Screen Addons

AEWindows	204
C Worthy w/forms	259
Curses-by Aspen Scientific	125
Greenleaf DataWindows	289
Hi-Screen XL	124
JAM by JYACC	464
VC Screen-painter	114
Vitamin C-source, menus	164

Utilities

BACK-IT	120
Copy II PC	35
Disk Technician Advanced	149
FASTBACK Plus	142
HELP ME	90
MACE GOLD	129
MACE Utilities	90
Norton Utilities Advanced	101
PC/Tools Deluxe 5.5	115
Vfeature Deluxe	111
V opt	111
XENOCOPY-PC	70
XTree Pro	111

Miscellaneous

Other Products

Aspen Korn Shell	109
CO/SESSION	214
dQuery/Lib	274
Evertrak	279
FM-Plus	84
Link & Locate + + − ROM MSC	339
Math Advantage	454
Norton Guides	75
OPTune-disk optimizer	84
Smartern 240	294
Zap-file xfer	74

Case & Prototypes

Dan Bricklin Demo II	159
Grasp	127
Instant Replay-Nostradamus	134
Interactive EasyFlow	119
MetaDesign by Meta Software	329
Pro-C	565
PROTEUS Prototype System	124
Show Partner F/X-demos	315
Softdemo	59
Visable Analyst	579

Tools

Debuggers

386 DEBUG	164
Optdebug	109
Periscope 0K	465
Sherlock	179
Softprobe 86/PX	345

Development Tools

Codesifter	85
DASM	239
Inside	109
MKS AWK	84
MKS Lex & Yacc	205
MKS RCS	155
PC-Lint	104
Pfinish	229
Plink 86 Plus-overlays	399
PolyMake	129
PVCS Corporate	355
.RTLINK-by Pocket Soft	184

Editors

BRIEF	Call
w/dBRIEF	Call
Ed, The Programmer's Edition	315
EDIX	155
Epsilon	155
KEDIT	119
Me w/source	169
MKS Vi	125
Multi-Edit	90
Norton Editor	55
NROFF/PC	117
PC/EDT +	269
Pi Editor	129
Qedit	79
SLICK Editor	170
SPF/PC	189
VEDIT PLUS	115
Vq	Call

OS

Control Programs

Concurrent DOS 386 (3users)	335
QEMM 386	55
PC-MOS 386 (single user)	179
VM/386	189

OS-Support

DESQview	110
MS Windows/286	69
MS Windows/386	144
Development Toolkit	319

386 Products

386 Max Pro	115
High C 386	835
Paseal 386	799

OS/2 Software

Btrieve	475
C tree	309
C worthy	339
Dataflex	1479
Greenleaf Data Windows	319
MKS Awk	159
MKS Lex & Yacc	339
MKS toolkit	399
OS/2 PM tool kit	359
Paradox	549
Power Search	135
Small talk/v PM	460
VC Screen	139
Vitamic C	289
Xtrieve Plus	475

1-800-323-1809

PROGRAMMERS WAREHOUSE

7819 East Greenway Suite 8 • Scottsdale, AZ 85260 • Outside US call (602) 443-0580

All Prices and Offers Subject to Change Without Notice

Listing Two *(Listing continued, text begins on page 16.)*

```
                E=W[h]->energy(*M1);
                if(A.length()==W[h]->width())
                        M2=new matrix(XY[h].a,A);
                else
                        M2=new matrix(A,XY[h].b);
                if (  ( E-(W[h]->depth()*W[h]->width()) < emin )
                   && (E==W[h]->energy(*M2))
                )
                {
                        emin=E-(W[h]->depth()*W[h]->width());
                        minimum=h;
                }
                delete M1;
                delete M2;
        }
        return XY[minimum];
}
void bam_system::train(char *patternfile)
// A "multiple-pair" encode - which Simpson calls "encode"
// this could be used for initial BAM Sys training. However an up
// and running BAM Sys should only need to use "encode".
{
        FILE *f=fopen(patternfile,"r");int n=0;
        filebuf sfile(f);
        istream s(&sfile,0);
        vecpair AB;
        for(;;){
                s >> AB;
                if(s.eof())break;
                D(cout << "Encoding " << n++ << "-th pattern pair:\n" << AB;)
                encode(AB);
        }
        D(cout << "Completed training from " << patternfile;)
}
ostream& operator<<(ostream& s,bam_system& b)
// operator to print out contents of entire BAM system
{
        for(int i=0;i<b.M;i++)
                s<< "BAM Matrix " << i << ": \n" << *(b.W[i]) << "\n";
}
```

End Listing Two

Listing Three

```
/////////////////////////
// TESTBAM.HPP
// Interactive BAM System Demonstration Program. Used to verify BAM system
// algorithms and demonstrate them on an abstract (i.e. just 0s and 1s) case.
// Developed with Zortech C++ 2.0 -- Copyright (c) 1989,90 Adam Blum
#include"bam.hpp"

vec v;
vecpair AB;
bam_system B;
char *p;
char patternfile[16]="TEST.FIL"; // file where test data is stored
int trace=0; // SET TRACE=<whatever> at DOS prompt to turn trace on
main()
{
     cout << "Interactive BAM System Demonstration\n";
     trace=(p=getenv("TRACE"))?1:0;
     cout << "Training from " << patternfile << "\n";
     B.train(patternfile);
     D(cout << "Resulting BAM System\n" << B;)
     cout <<"Enter patterns as 0's and 1's terminated by comma.\n"
     <<"Patterns must be length of " << ROWS << " or " << COLS <<".\n"
     << "Null vector (just "","") to end.\n\n" ;
     for(;;){
          cout << "Enter pattern: ";
          cin >> v;
          if(!v.length())break;
          if(v.length()!=ROWS && v.length()!=COLS){
               cout << "Wrong length.\n";
               continue;
          }
          AB=B.recall(v);
          cout << "Recalled pattern pair\n" << AB;
     }
}
```

End Listing Three

Listing Four

```
    1100101011010011,11101010,
    0110110111110110,11010101,
    1101111001010101,11110010,
    1010101000010111,11001101,
    0011001101011011,11110100,
    1100101011010011,11101010,
    0110100111110110,11010101,
    1101110101010101,11110010,
    1011101010010111,11001101,
    0001011101011011,11110100,
    1100101001010011,11101010,
    0110110110110110,11010101,
    1100111011010101,11110011,
    1010000100010111,11001101,
    0001101101011011,11110110,
    1100100011010011,11100110,
    0110110011110110,11010101,
    1101111001010101,11110011,
    1010100000011111,11001101,
    0001100101111011,11111000,
    1100101011010011,11011010,
    0010100111110110,11010101,
    1101111101010101,11110010,
    1010111000010111,11101101,
    0001000001011011,11110100,
    1100101011010011,11101010,
    0110110111110110,11010101,
    1101111000010101,11110110,
    1010100111010111,11001101,
    0001000101011011,11110100,
    0110110101110110,11010111,
    1101111001010101,11110110,
    1010111100110111,11001101,
    0001000101011011,11110100,
    1100101010010011,11101010,
    0110110111110110,11010101,
    1101111001010101,11110010,
    1010110000010111,11001101,
    0011000101011011,11110100,
    0011010101111011,10010111,
```

End Listing Four

Listing Five

```
# TESTBAM.MK
# Make file for BAM System implementation tester
# Uses Microsoft Make
# Compiler: Zortech C++ 2.0
# To make with diagnostics enabled:
# make CFLAGS="-DDEBUG=1" testbam.mk
#

CFLAGS=
.cpp.obj:
     ztc -c $(CFLAGS) $*.cpp
bam.obj: bam.cpp bam.hpp
testbam.obj: testbam.cpp bam.hpp
testbam.exe: testbam.obj bam.obj
     blink testbam bam;
```

End Listings

Announcing!

bStrings

Dynamic String Handling For C TM

Since 1972, C programmers have waited for *true dynamic string handling* facilities like those in BASIC.

bStrings : Variable length string handling that is automatic and fully transparent. Save development time. Concentrate on your program - not on string manipulations.

The ***bStrings*** library is coded for efficiency - using memory management techniques guaranteed not to fragment heap space.

$89.95

30 day unconditional guarantee.

Available for Microsoft C, Turbo C or Quick C.

Visa/MC/Amex/C.O.D.

TRUE DYNAMIC STRINGS

Over 100 Functions Covering:

* Cut * Copy * Paste * Concatenate
* MidStr * User Input * Search * Replace
* Buffered file I/O * Low Level file I/O

No royalties ever. * Source code available. * Lots of examples.

COMING SOON -*TL4C: The Text Library for C.*
The Worlds Most Comprehensive Text Library For C Programmers

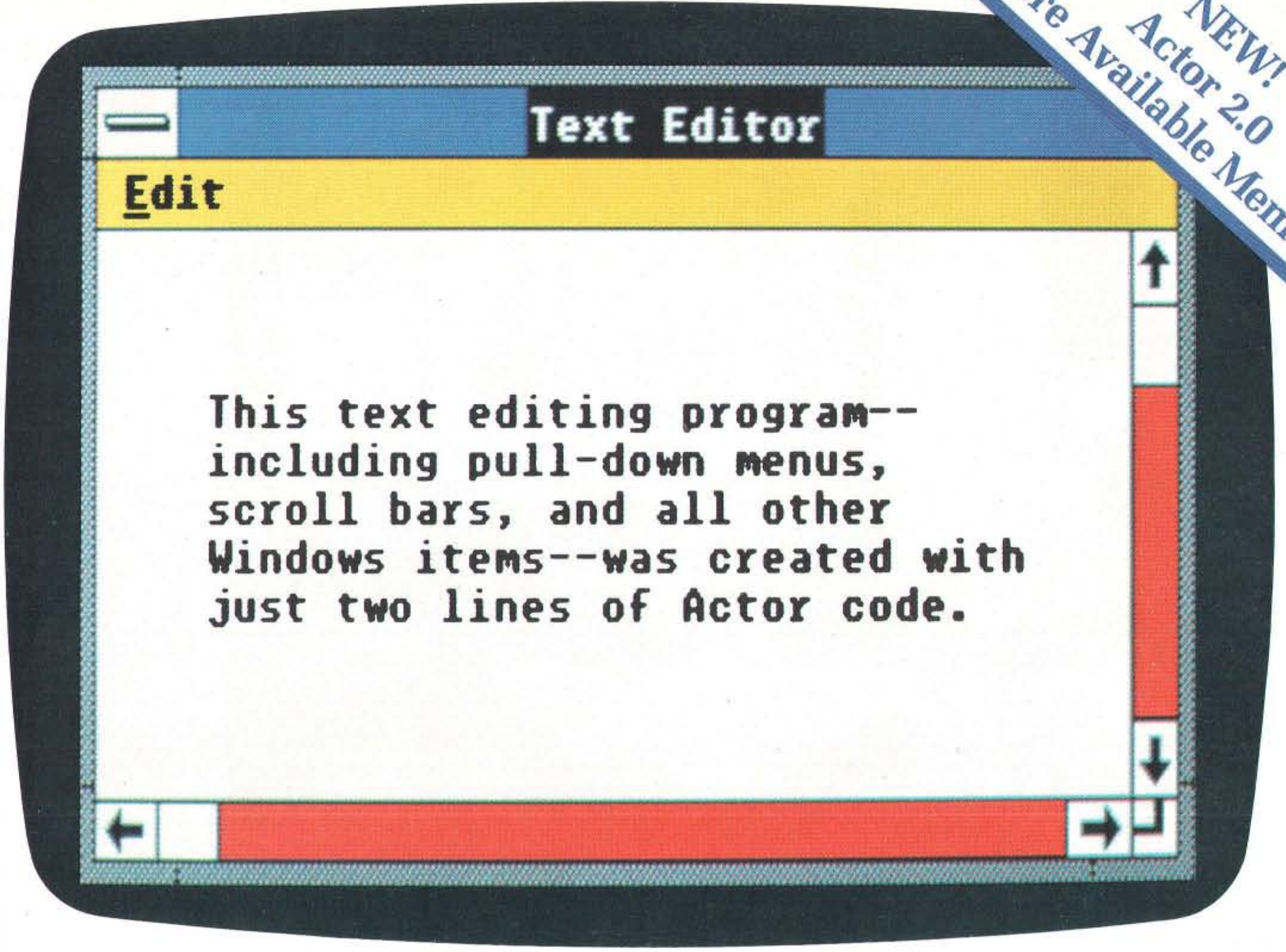
Robust Text Buffer Management * Hypertext * Real Unix regular expressions *Interfaces to virtually all screen management packages. Users upgrading to the full TL4C library wil receive credit for the full purchase price of bStrings. Trademarks are the property of their respective owners.

KBM Communications, Inc.
2401 Lake Park Dr. Ste. 160 Atl., Ga. 30080

1-800-277-0303

CIRCLE NO. 118 ON READER SERVICE CARD

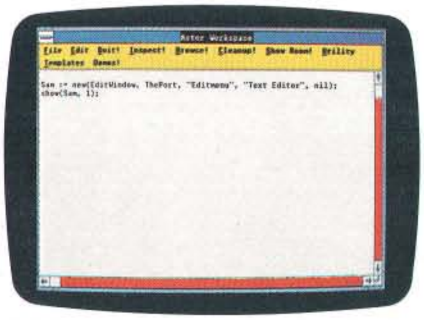
NEW!
Actor 2.0
More Available Memory



This text editing program--including pull-down menus, scroll bars, and all other Windows items--was created with just two lines of Actor code.

Actor™ is the fastest, easiest way to develop applications for Microsoft® Windows. The reason—Actor's reusable toolkit of objects such as dialog boxes and edit windows. It more than doubles your overall productivity, making Actor an essential part of any Windows development project.

A full-featured, interactive, Windows-based programming environment, Actor provides immediate compilation, interactive testing, and source-code debugging. You can use it to produce fast standalone applications that support all Windows features, including Dynamic Data Exchange (DDE) and expanded memory.


Actor Workspace
File Edit Doit! Inspect! Browse! Cleanup! Show Room! Utility
Templates Demos!

```
Sam := new(EditWindow, ThePort, "Editmenu", "Text Editor", nil);
show(Sam, 1);
```

These two lines of Actor code are all it took to produce the Windows text editing program you see above. Just think what you can do with a few more lines.

Actor is a powerful, pure object-oriented programming language. It's all you need to develop complete Windows applications. You also have the option to dynamically link Microsoft C code to your Actor program.

Either way, it's the fastest way to produce everything from prototypes to complete development projects. No wonder so many developers are already using Actor.

Call us now for more information.

The sooner you do, the sooner you can speed up *your* Windows development work.

The Whitewater Group®

600 Davis Street
Evanston, Illinois 60201 U.S.A.
(708) 328-3800
FAX (708) 328-9386

(800) 869-1144

Two New Products
For C or Actor Programmers

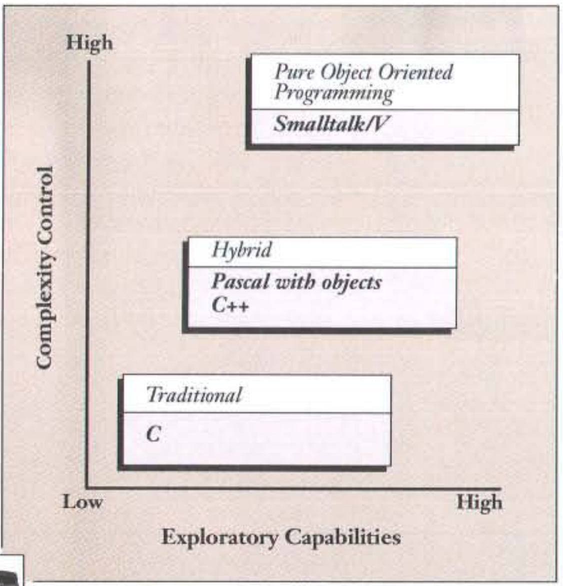
WinTrieve™
ISAM indexed file manager. Only $395, no royalties.

Whitewater Resource Toolkit™
Edit dialog boxes, bitmaps, icons and more. Only $195.

Smalltalk/V.® Designed to blow the doors off the hybrid languages of the programming world. Smalltalk/V does prototyping the same way Shelby prototyped the Cobra... using a blend of technical expertise and seat-of-the-

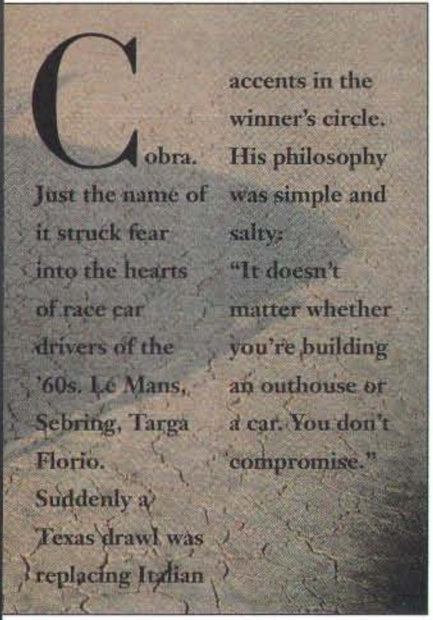
"Anyone can build a prototype by the by the seat of your pants." — CARROLL SHELBY *Creator of the legendary Shelby Cobra*

pants savvy that's startlingly sophisticated. First you doodle, design, dream. Then you explore the possibilities and begin to assemble the prototype. You test. You tinker. You change. And you keep on changing and test-driving and refining until the prototype is just the way it was meant to be. With no compromises... of any kind. But the most remarkable thing is this prototype is not just a prototype. It runs, it races, it performs like the real application. Because it is the real application. And you achieve this feat without once having to go through the old "crash and burn" kind of programming so common with languages born in the age of mainframes.



COMPLEXITY CONTROL FOR THE 1990s AND BEYOND.

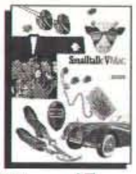
The concept behind an object-oriented programming system is relatively simple. You build more complex objects out of simpler ones. Much as you can build complicated designs with a Lego set. With Smalltalk/V a programmer can write a piece of code and then



Cobra. Just the name of it struck fear into the hearts of race car drivers of the '60s. Le Mans, Sebring, Targa Florio. Suddenly a Texas drawl was replacing Italian accents in the winner's circle. His philosophy was simple and salty: "It doesn't matter whether you're building an outhouse or a car. You don't compromise."

THE "AM I READY FOR SMALLTALK/V" CHECKLIST

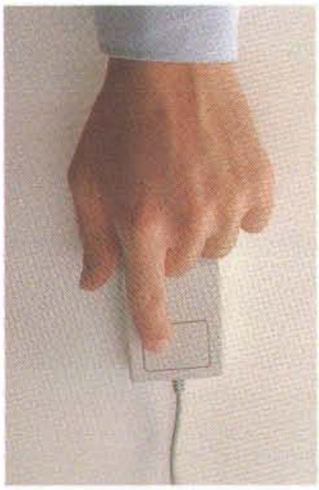
- ☐ Does a lot of your work involve prototyping/exploratory programming?
- ☐ Are many of your problems difficult to define?
- ☐ Are external factors constantly changing?
- ☐ Do you like to make changes from insights gathered along the way?
- ☐ Do you feel torn between efficiency and conceptual clarity?
- ☐ Are you developing for MultiFinder or Presentation Manager?
- ☐ Are you tired of needless crashing?
- ☐ Are team projects getting harder to manage and complete on time?
- ☐ Has your creativity been intimidated by the rigorous demands of the process?



"Traditional computer languages and interfaces with their structure and detail, have appealed to those of us who are left-brained (more logical and analytical). On the other hand, object-oriented languages and interfaces, with their emphasis on perception and the whole picture, invite those of us who are right-brained (more artistic and intuitive) to join the computer revolution as well."

—Byte

book. But you create a legend



With Smalltalk/V your mouse becomes a hot programming tool for either your Mac or your PC. You'll find that Smalltalk/V is souped up with lots of other high performance features, too. The Class Hierarchy Browser, Inspector, Debugger, Class Browser, Method Browser and Walkback window are all standard equipment.

reuse it again and again. The "inheritance" factor lets you create, enhance and refine your applications without constantly having to re-invent the wheel. Or, as one programmer put it, "With Smalltalk/V you can write a fugue without having to build the piano."

OOPS! LOOK WHAT THE WORLD IS COMING TO.

"The software of the future, OOP promises not only to boost programmer productivity but also put powerful computing capabilities in the hands of non-techies."

—Business Week

"Object-oriented programming is the key to the next great transition in personal computing."

—NY Times

AT THESE PRICES IT'S CERTAINLY NOT MONEY THAT'S HOLDING YOU BACK.

Smalltalk/V (DOS 512K RAM)	**$99.95**
Smalltalk/V 286 (286 or 386 1.5 MB RAM)	**199.95**
Smalltalk/V Mac (Plus, SE, II 1.5 MB RAM)	**199.95**

Smalltalk/V. A product of Digitalk Inc., 9841 Airport Blvd., Los Angeles, CA 90045. For information or to find a dealer near you call:

1-800-922-8255
1-213-645-1082
CompuServe 71361,1636

Smalltalk/V

MultiFinder is a trademark of Apple Computer. Smalltalk/V is a registered trademark of Digitalk Inc. Prices subject to change without notice.

CIRCLE NO. 115 ON READER SERVICE CARD

Now you can bring the power of DESQview to Clipper,™ dBASE™ and Basic.

Quarterdeck can provide everything you need to put DESQview capabilities into your applications.

There's a growing realization among programmers that for many users, the ideal PC of the future doesn't look all that different from the PC of today. Experts agree that extended DOS will be the dominant operating system for the foreseeable future.

And DESQview is the best way to create powerful, multitasking solutions for the 30 million DOS PCs in use today.

The API Advantage

Programmers who take advantage of DESQview's API (Application Program Interface) get easy access to the powerful capabilities built into DESQview—multitasking, windowing, intertask communications, mailboxes, shared programs, memory management, data transfer, mousing, menu-building and context-sensitive help. In short, you get all the important benefits promised by OS/2. Without the overhead burden and performance trade-offs.

API capabilities are now available to Clipper, Basic and dBASE users as well as Pascal and C programmers.

Bells and Whistles

A program taking advantage of the DESQview API can spawn subtasks for performing background operations or new processes for loading and running other programs concurrently. It can schedule processing after an interval or at a certain time. It can use DESQview's intertask communications to rapidly exchange data between programs, share common code and data; or interrupt at critical events. It can use DESQview's menuing and mousing capabilities to create menus. And there's lots more it can do.

80386 Power

Programmers can take advantage of the 80386's protected mode for large programs, yet run on DOS and multitask in DESQview—side by side with other 80386 and DOS programs. The breakthroughs that make it possible: DOS Extenders from PharLap, Rational Systems, Eclipse and Oracle and DESQview support of these extenders.

A Complete Interactive Toolkit —Including a Debugger

API Reference Manual The key to the power of the DESQview API, this manual contains all you need to know to write Assembly Language programs that take full advantage of DESQview's capabilities. And there's an 'include' file with symbols and macros to aid you in development.

Interactive tools like this Panel Designer make implementing the DESQview API a snap.

API Clipper Library Add new power to client-server database applications. Define tasks that monitor or control any other application; run reports, sorts or numerical analyses in the foreground or background; create custom interfaces with control over keyboard, mouse and window functions. And it's all easy to accomplish because you can treat the entire set of over 300 object-oriented API functions as prewritten UDFs.

"With the DESQview API, you can create programs with abilities far beyond those of Clipper alone... obviously, the uses are limited by your imagination"
—Data Based Advisor, December 1989

You get the API Reference Manual and source code for the library with the Clipper Library package.

API dBASE Library All API functions are available within dBASE III and IV. We include source code for the library, the API Reference Manual and example programs.

API Basic Library The Basic library provides interfaces for the entire set of API functions. Included are the API Reference Manual, source code for the library and example programs.

API Pascal Library The Pascal library provides interfaces for all API functions. It supports Borland Turbo Pascal 4.0 and 5.0 compilers and Microsoft Quick Pascal Included are the API Reference Manual, source code for the library and example programs.

API C Library You get C language interfaces for the entire set of API functions. It supports the Lattice™ C, Metaware™ C, Watcom C, Zortech C, C++, Microsoft® C, and Turbo C compilers for all memory models. Included with the C Library package is the API Reference Manual and source code for the library.

API Debugger The DESQview API Debugger is an interactive tool enabling the API programmer to trace and single step through API calls from several concurrently running DESQview-specific programs. Trace information is reported symbolically along with the program counter,registers, and stack at the time of the call. Trace conditions can be set to report only calls of interest

API Panel Designer This interactive tool helps you design windows, menus, help screens, error messages, and forms. It includes an editor that lets you build an image of your panel using simple commands to enter, edit, copy, and move text, as well as draw lines and boxes. You can then define the characteristics of the window that will contain the panel: its position, size, and title. Finally, you can specify the locations and types of fields in the panel.

The Panel Designer automatically generates all the DESQview API data streams necessary to display and take input from your panel. These data streams may be grouped into panel libraries and stored on disk or as part of your program.

Some of the applications under development right now using DESQview API Tools: CAD, medical systems, insurance, 3270 network management, mainframe communications, real estate management, typesetting, point of sale, education, stock and commodity trading and online voting.

More Tools are Coming If you'd like to know all the things we're up to in the API Toolkit, plan to attend our API Conference this Summer. Call or write for details.

But for now, we can tell you that we are committed to adding tools as needed by our users. In the works, we have many new, exciting tools that will keep DESQview API users at the forefront of interface development.

Quarterdeck

Quarterdeck Office Systems, 150 Pico Blvd., Santa Monica, CA 90405 (213) 392-9851 Fax: (213) 399-3802

CIRCLE NO. 204 ON READER SERVICE CARD

©1990 Quarterdeck. Trademarks are property of their respective owners.

Trademarks: Clipper, Nantucket; dBASE, Ashton-Tate; OS/2, IBM; PharLap, Rational Systems, Eclipse, Eclipse; Oracle, Lattice, Metaware, Watcom, Zortech, Microsoft, Turbo C

©1990 Quarterdeck Office Systems

Listing One *(Text begins on page 28.)*

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: andy; Base: 10 -*-

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;       A Research Environment for the Instantiation of Neural Networks
;;;       Andrew  J. Czuchry, Jr. -- Georgia Institute of Technology
;;;       Georgia Tech Research Institute -- Artificial Intelligence Branch
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;             Knowledge Representation Structure definitions
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defstruct (NET                                ; structure for a network
            (:print-function net-printer))     ; printer function
   layers                      ; list of layers
   local-parameters            ; list of local parameters for net
  )
(defstruct (LAYER                              ; structure for a layer
            (:print-function layer-printer))   ; printer function
  planes                      ; list of planes
  e-connections               ; list of offsets for excitatory connections
  i-connections               ; list of offsets for inhibitory connections
  local-parameters            ; list of local parameters for layer
  prev-layer                  ; "ptr" to preceding layer structure
  type                        ; layer type (e.g., "S" or "C" in
                              ;  neocognitron; "F1" or "F2" in ART)
  )
(defstruct (PLANE             ; structure for a plane (sub-network)
            (:print-function plane-printer))  ; printer function
  (cells nil :type array)     ; cell values for plane
  e-connections               ; list of offsets for excitatory connections
  i-connections               ; list of offsets for inhibitory connections
  e-weights                   ; list of excitatory weights [real]
                              ;  (same order as list of connections)
  i-weights                   ; list of inhibitory weights [real]
                              ;  (same order as list of connections)
  size                        ; size of plane (list N x M)
  local-parameters            ; list of local parameters for plane
  layer                       ; "ptr" back to layer of which plane is a part
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;     Structure Printer Functions -- Written by Harold S. Forbes

;;; -----------------------------------------------------------------------
;;; The function OBJECT-ADDRESS gets the memory address of any LISP object.

(defun OBJECT-ADDRESS (object)
  ;; Symbolics implementation.
  (sys:%pointer object)
  )
(defun NET-PRINTER (structure stream ignore)
  (declare (ignore ignore))
  (format stream "#<net ~X>" (object-address structure)))
(defun LAYER-PRINTER (structure stream ignore)
  (declare (ignore ignore))
  (format stream "#<~A-layer ~X>" (layer-type structure)
          (object-address structure)))
(defun PLANE-PRINTER (structure stream ignore)
  (declare (ignore ignore))
  (format stream "#<plane ~X>" (object-address structure)))
```

End Listing One

Listing Two

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: andy; Base: 10 -*-

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;       A Research Environment for the Instantiation of Neural Networks
;;;       Andrew  J. Czuchry, Jr. -- Georgia Institute of Technology
;;;       Georgia Tech Research Institute -- Artificial Intelligence Branch
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                       Net CREATION functions
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Create a new net.
;   Creates a version of a 5-layer neocognitron by default.

(defun CREATE-NET (&key
                   (num-of-layers 5)
                   (num-of-planes-per-layer-list '(1 24 24 24 24))
                   (plane-size-list '((16 16) (16 16) (10 10) (4 4)
                                      (1 1)))
                   (connection-pattern 'square)
                   (mask-size-list '((5 5) (5 5) (5 5) (2 2)))
                   (net-parameters '(:  0.5))
                   (additional-parameter-list
                    '(:net-type neocognitron
                                (:r-val-list '(4.0 1.5 1.5))
                                (:q-val-list '(1.0 16.0 16.0))
                                (:b-val-list '(0.0 0.0 0.0))
                                (:orientation-list '(8 1 8 1))
                                )))
  (let* ((layer-list
         (create-layer-list num-of-layers
                            num-of-planes-per-layer-list
                            plane-size-list
                            connection-pattern
                            mask-size-list
                            additional-parameter-list))  ; create layers
                                                         ;  meeting specification
                                                         ;  parameters
        (net (make-net :local-parameters net-parameters
                       :layers layer-list)))   ; create knowledge structure
                                                         ; storing new net
    net)
  )

;; Create a list of NUM-OF-LAYERS layers of the approriate type (type
;;   key recorded in ADDITIONAL-PARAMETER-LIST).
(defun CREATE-LAYER-LIST (num-of-layers planes-per-layer-list
                         plane-size-list con-pattern
                         mask-size-list
                         additional-parameter-list)
  (let ((layer-type (zl:lexpr-funcall #'extract-type-key
                                      additional-parameter-list)))
; determine type of net to which layers are to belong
;  and create appropriate type of layers
    (zl:selectq  layer-type
      (neocognitron
        (create-neocognitron-layer-list
          num-of-layers planes-per-layer-list
          plane-size-list con-pattern mask-size-list
          additional-parameter-list))
      (ART2
        (create-ART2-layer-list
          num-of-layers planes-per-layer-list
          plane-size-list con-pattern mask-size-list
          additional-parameter-list))
      (backpropagation
        (create-backprop-layer-list
          num-of-layers planes-per-layer-list
          plane-size-list con-pattern mask-size-list
          additional-parameter-list))
      )
    )
  )

;; Create a list of NUM-OF-LAYERS layers for the neocognitron
(defun CREATE-NEOCOGNITRON-LAYER-LIST (num-of-layers planes-per-layer-list
                                                     plane-size-list
                                                     con-pattern
                                                     mask-size-list
                                                     additional-parameter-list)
; extract parameters specific to neocognitron
  (let* ((r-val-list (zl:lexpr-funcall #'extract-r-val-list
                                       additional-parameter-list))
         (q-val-list (zl:lexpr-funcall #'extract-q-val-list
                                       additional-parameter-list))
         (b-val-list (zl:lexpr-funcall #'extract-b-val-list
                                       additional-parameter-list))
         (orientation-list (zl:lexpr-funcall #'extract-orientation-list
                             additional-parameter-list))
         (number-of-processing-layers
          (- num-of-layers 1))
         (num-of-s-layers (/ number-of-processing-layers 2)))
; error checking and layer creation
    (cond ((not (= num-of-s-layers (length r-val-list)
                   (length q-val-list) (length b-val-list)))
                                              ; check extracted parameters
           (ferror "Improper parameters for a net with ~D S-layers:
                     r value list = ~s,
                     q value list = ~s,
                     b value list = ~s."
                   num-of-s-layers r-val-list q-val-list b-val-list))
          ((not (= num-of-layers
                   (length planes-per-layer-list)
                   (length plane-size-list)))        ; check passed parameters
           (ferror "Improper parameters for a net with ~D layers:
                     Either not enough planes sizes listed in ~s, OR
                      not enough plane sizes listed in ~s."
                   num-of-layers planes-per-layer-list
                   plane-size-list))
          ((not (= number-of-processing-layers (length mask-size-list)))
                                                    ; check projection masks
           (ferror "Improper parameters for a net with ~D layers beyond
                   input layer:
                      Not enough connection mask sizes listed in ~s."
                   number-of-processing-layers mask-size-list))
; Create appropriate number of layers, one at a time, and record as a list.
; For each layer, extract appropriate parameter settings and sizes.
          (t
           (do* ((i 1 (+ i 1))
                 (r-val-list r-val-list (cdr r-val-list))
                 (r-val (car r-val-list) (car r-val-list))
                 (q-val-list q-val-list (cdr q-val-list))
                 (q-val (car q-val-list) (car q-val-list))
                 (b-val-list b-val-list (cdr b-val-list))
                 (b-val (car b-val-list) (car b-val-list))
                 (rest-orientations orientation-list
                                    (cddr rest-orientations))
                 (s-orientations (car rest-orientations)
                                 (car rest-orientations))
                 (c-orientations (cadr rest-orientations)
                                 (cadr rest-orientations))
                 (prev-plane-num (car planes-per-layer-list)
                                 (cadr planes-per-layer))
                 (planes-per-layer (cdr planes-per-layer-list)
                                   (cddr planes-per-layer))
                 (num-of-s-planes (car planes-per-layer)
                                  (car planes-per-layer))
                 (mask-list mask-size-list (cddr mask-list))
                 (mask-size (car mask-list) (car mask-list))
                 (plane-sizes-list (cdr plane-size-list)
                                   (cddr plane-sizes-list))
                 (prev-c-plane-size (car plane-size-list)
                                    c-plane-size)
                 (s-plane-size (car plane-sizes-list)
                               (car plane-sizes-list))
                 (c-plane-size (cadr plane-sizes-list)
                               (cadr plane-sizes-list))
; create input layer
                 (input-layer
                  (make-layer :planes
```

(continued on page 95)

epsilon has what you've been missing. . .

programmer's editor

Epsilon doesn't implement its features halfway. When we came out with Undo, we remembered Redo. Our extension language doesn't just borrow a few keywords from C, it has C's syntax and expressiveness. Maybe that's why Epsilon won a BYTE Award of Distinction. Compare Epsilon to your editor, and see what you're missing. . .

1 9 8 8 BYTE AWARD OF DISTINCTION

	epsilon	Brief
Undo and Redo	Complete Undo AND Redo! Any sequence of edits you make can be undone, one at a time, with the Undo key. You can even undo your undo's with the Redo key. That's very important: it lets you casually use the Undo facility.	Its "Complete Undo" does not let you redo. So be careful with that Undo key, because it's irreversible. Once you undo something, it's gone forever!
Extension Language	Our extension language, EEL, gives you the syntax of C and a rich set of data types. All commands are written in EEL, and you get source code to all of them. Best of all, EEL is fast!	Its "macro" language has a very limited set of data structures (unlike C or EEL). Most of Brief's standard commands were not written in its macro language. Why not?
Compiler Journaling	When you run another program inside Epsilon, it can first shrink to 10K, then run your program. You see your compiler run, see the error messages, etc. You can even give the program input. And when your program finishes, you're put back into the editor, and a complete input/output typescript is there in an Epsilon buffer!	It can shrink and run your program, but it uses command line redirection to gather the output of your program. This means that you COMPILE BLIND! If there are many error messages, you wait until the compiler finally exits to see ANY of them!
Multitasking Compiles	Run utilities, compilers, even the DOS command processor concurrently, while you continue to edit. If your program is very large, you can use Compiler Journaling.	It has no comparable multitasking facility.
EMS Support	Epsilon makes extensive use of EMS memory, if you have it. It uses EMS both to store what you're editing, and to store itself when you swap Epsilon out with a keystroke.	It will use EMS memory only to swap itself out, not to store your buffers. So, you're more cramped for memory than you should be.

Trademarks: "Epsilon" and "Lugaru" are trademarks of Lugaru Software Ltd. "Brief" is a trademark of Underware, Inc. Information on Brief is from the manual and conversations with their technical support staff. Versions: Epsilon 4.0, Brief 3.0.

Epsilon is available for DOS, OS/2, Xenix & 386/ix. Here's a partial list of features:

C-like extension language, called EEL
Source code to all commands
Convenient multilevel undo plus total redo
Scans compiler errors (customizable)
Swaps out of memory for big compilers
Unlimited file size, number of files
On-line tutorial for quick startup
As many windows as will fit on the screen
Supports large displays (like VGA & EGA)
Context sensitive help autoconfigures
Tags package indexes subroutine definitions
Files can be larger than available memory
Fully utilizes EMS memory
Interactively customizable keyboard
EMACS-style command set
Record keystroke sequences
Source level debugger & profiler for EEL
Do interrupts from EEL (DOS)
Call any dynamic-link library for EEL (OS/2)
Saves deleted text (n times)
Navigate through your directory structure
Not copy protected, for your convenience
Multitasks compilers, even under DOS
Language support (C, Lisp, etc.)
Completion on file names, commands, tags, ...
Full command line editing
Optional backups
Regular expression search and tagged replace

Ask about our free trial!
$195 direct
60-day money-back guarantee!
Orders & Info: (412) 421-5911
FAX: 412.421.6371

Lugaru Software Ltd. / 5843 Forbes Avenue / Pittsburgh, PA 15217

CIRCLE NO. 144 ON READER SERVICE CARD

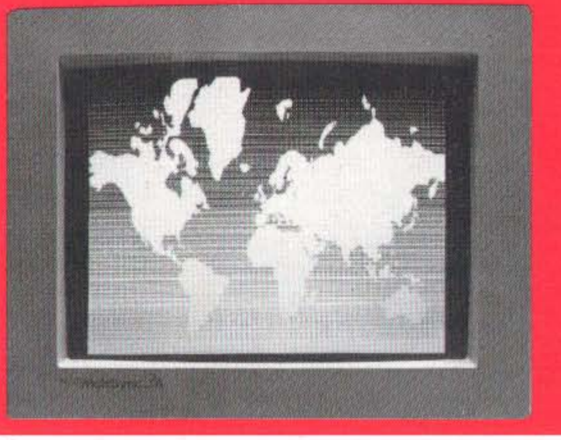
Listing Two *(Listing continued, text begins on page 28.)*

```
                                     (create-plane-list
                                      1 prev-c-plane-size
                                      (do ((i 1 (+ i 1))
                                           (times (apply #'* prev-c-plane-size))
                                           (res '((0)) (cons '(0) res)))
                                          ((>= i times) res))
                                      0 0 1 mask-size 'C)
                                     :type 'C))
; create connections
                  (s-connection-list
                   (create-connection-list 1 prev-c-plane-size s-plane-size
                                           con-pattern mask-size 'S)
                                      ; Connections same for for all planes
                   (create-connection-list 1 prev-c-plane-size s-plane-size
                                           con-pattern mask-size 'S))
                  (c-connection-list
                   (create-connection-list 1 s-plane-size c-plane-size
                                           con-pattern (cadr mask-list) 'C)
                                         ;C-cells connect one S-plane
                   (create-connection-list 1 s-plane-size c-plane-size
                                           con-pattern (cadr mask-list) 'C))
; create planes
    (s-planes
      (create-plane-list num-of-s-planes s-plane-size s-connection-list
                         prev-plane-num b-val s-orientations mask-size 'S)
      (create-plane-list num-of-s-planes s-plane-size s-connection-list
                         prev-plane-num b-val s-orientations mask-size 'S))
    (c-planes
     (create-plane-list (cadr planes-per-layer) c-plane-size c-connection-list
                         num-of-s-planes b-val c-orientations mask-size 'C)
     (create-plane-list (cadr planes-per-layer) c-plane-size c-connection-list
                         num-of-s-planes b-val c-orientations mask-size 'C))
;assign layers
    (new-s-layer (make-layer :planes s-planes :e-connections s-connection-list
                :local-parameters '(:net-type neocognitron :r ,r-val :q ,q-val)
                                        :prev-layer input-layer :type 'S)
                  (make-layer :planes s-planes :e-connections s-connection-list
                :local-parameters '(:net-type neocognitron :r ,r-val :q ,q-val)
                                         :prev-layer (car (last layers))
                                         :type 'S))
    (new-c-layer (make-layer :planes c-planes :e-connections c-connection-list
                 :local-parameters '(:net-type neocognitron :r ,r-val :q ,q-val)
                                         :prev-layer new-s-layer :type 'C)
                  (make-layer :planes c-planes :e-connections c-connection-list
                 :local-parameters '(:net-type neocognitron :r ,r-val :q ,q-val)
                                         :prev-layer new-s-layer :type 'C))
; add new layers to layer list
                  (layers                     ; S and C layers
                   (list input-layer new-s-layer new-c-layer)
                   (append layers
                          (list new-s-layer new-c-layer))))
                 ((>= i num-of-s-layers) layers)))   ; return list of layers
         )
     )
  )

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                  Keyword and Parameter EXTRACTION functions

(defun EXTRACT-TYPE-KEY (&key net-type &allow-other-keys)
  net-type)                               ;; Extract the NET-TYPE keyed value
(defun EXTRACT-R-VAL-LIST (&key r-val-list &allow-other-keys)
  r-val-list)                             ;; Extract the R-VAL-LIST keyed value
(defun EXTRACT-Q-VAL-LIST (&key q-val-list &allow-other-keys)
  q-val-list)                             ;; Extract the Q-VAL-LIST keyed value
(defun EXTRACT-B-VAL-LIST (&key b-val-list &allow-other-keys)
  b-val-list)                             ;; Extract the B-VAL-LIST keyed value
(defun EXTRACT-ORIENTATION-LIST (&key orientation-list &allow-other-keys)
  orientation-list)                     ;; Extract the ORIENTATION-LIST keyed value
(defun EXTRACT-NET-  (net)              ;; Extract the   parameter from a NET
  (zl:lexpr-funcall #'extract-net- -aux
                    (net-local-parameters net)))
(defun EXTRACT-NET- -AUX (&key   &allow-other-keys)
   )                                   ;; Extract the   keyed value
(defun EXTRACT-LAYER-R (layer)          ;; Extract the R parameter from a LAYER
  (zl:lexpr-funcall #'extract-layer-r-aux
                    (layer-local-parameters layer)))
(defun EXTRACT-LAYER-R-AUX (&key r &allow-other-keys)
  r)                                    ;; Extract the R keyed value
(defun EXTRACT-LAYER-Q (layer)          ;; Extract the Q parameter from a LAYER
  (zl:lexpr-funcall #'extract-layer-q-aux
                    (layer-local-parameters layer)))
(defun EXTRACT-LAYER-Q-AUX (&key q &allow-other-keys)
  q)                                    ;; Extract the Q keyed value

 (defun EXTRACT-PLANE-B (plane)         ;; Extract the B parameter from a PLANE
  (zl:lexpr-funcall #'extract-plane-b-aux
                    (plane-local-parameters plane)))
 (defun EXTRACT-PLANE-B-AUX (&key b &allow-other-keys)
  b)                                    ;; Extract the B keyed value

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                  Keyword and Parameter INSERTION functions

 (defun INSERT-KEYED-VAL (old-list insertion-list)  ;; Insert a keyed value
  (do ((list (cddr old-list) (cddr list))
       (i-key (car insertion-list))
       (o-key (car old-list) (car list))
       (o-val (cadr old-list) (cadr list))
       (result insertion-list                      ; insert new keyed value
               (append result                          ; keep old values
                    (if (not (equal o-key i-key))      ;  for other keys
                        (list o-key o-val)))))
      ((null o-key) result)))
```

End Listing Two

(continued on page 96)

CUT PROGRAMMING TIME. WITH GRAPHIC RESULTS.



Get to the market quicker with graphics that don't compromise performance. GSS® Graphics Development Toolkits for DOS and OS/2 give you access to more than 100 high-level graphics functions. And your GDT-based applications will support over 300 graphics devices.

The GDT is a time-proven solution for programs written in C, FORTRAN, Pascal, BASIC Compiler or Macro Assembler. It has been optimized through eight years of refinement and application to produce high-quality, high-performance graphics.

YOU'LL BE IN GOOD COMPANY.

IBM has licensed the GDT for its three PC operating systems: DOS, OS/2 and AIX. SPC's Harvard Graphics and Ashton-Tate's Draw APPLAUSE use GDT technology. So do hundreds of other PC packages for science, business and engineering.

IN A HURRY?

CALL (503) 641-2455.

ASK FOR DEPARTMENT DD

Order your GDT, or ask for a free copy of our Devices Supported booklet. The GDT is the most direct route to completing your applications—with a graphic difference.

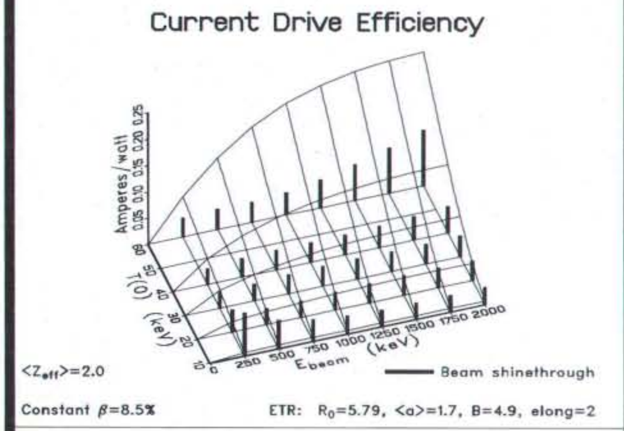
GSS®

Graphic Software Systems

CIRCLE NO. 66 ON READER SERVICE CARD

GraphiC™

***"gives you all the C language routines you need to write an impressive scientific graphing program of your own. Highly recommended"** – PC Magazine*



PC : $395 (source code) **Mac : $295 (no source)**

For *personal* use only.

Scientific Endeavors Corporation

508 N. Kentucky St.; Kingston, TN 37763

(615) 376-4146 **FAX: (615) 376-1571**

CIRCLE NO. 208 ON READER SERVICE CARD

MODULA-2

M2VMS™ Release 4.1-2

The Reliable Tool for Software Professionals

The Modula-2 development system for VAX by TERRA Datentechnik offers full flexibility within the VMS environment. M2VMS 4.1-2 is based on the well-known Modula-2 compiler by LOGITECH that has recently been aquired by TERRA Datentechnik.

Highlights

- *supports the VMS-Debugger*
- *supports LSE*
- *interfaces the operating system*
- *fully compatible with other languages*
- *Make Utility*
- *library as object and source code*
- *maintenance contract available*

Please ask for detailed information.
TERRA Datentechnik also offers a large choice of Modula-2 products for PCs.

TERRA Datentechnik Bahnhofstrasse 33
CH - 8703 Erlenbach Switzerland
Tel ++41 - 1 - 910 35 55 Fax ++41 - 1 - 910 19 92

CIRCLE NO. 426 ON READER SERVICE CARD

Listing Three *(Listing continued, text begins on page 28.)*

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: andy; Base: 10 -*-

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;      A Research Environment for the Instantiation of Neural Networks
;;;      Andrew  J. Czuchry, Jr. -- Georgia Institute of Technology
;;;      Georgia Tech Research Institute -- Artificial Intelligence Branch
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                          Net TRAINING functions
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Top level function for training.  Trains on all patterns in PATTERN-LIST.
;;   Sets mappings.
;     Trains on *neocognitron* by default.
(defun TRAIN-MAIN-LOOP (&key
         (net *neocognitron*)
         (pattern-list (mapcar #'item-misc (scl:send lrn-pattrns :item-list)))
         (iteration-list '(4 4 4 4))
        print-data)
(let* ((net-type (x):lexpr-funcall &"extract-type-key
                                  (net-locate-parameters net)))
      (iterations (compute-iteration-list net-type i iteration list)
      (actual-its (    iterations 1)))
  (do* ((i 0 (+ i 1))
     (conditioned-it-list  (select-iteration-list net-type i iteration-list)))
     (terminate-iterations net-type i iterations))
  (do* ((patterns pattern-list (cdr patterns))        ; loop over all patterns
       (pattern (car patterns) (car patterns))
       (actual-its (length pattern-list)))
      (cond (print-data
             (format t "~%~%")
             (time:print-current-time)
             (format t "~%% ** training ~s on pattern ~s.
                 (Iteration ~d of ~d, ~d more patterns.)"
                     net pattern i actual-its (length (cdr patterns)))))
      (train pattern net iteration-list)                    ; perform training
      )
)
  (create-mappings net pattern-list)        ; record mapping between pattern and
                                            ;  most active "output layer" cell
  (cond (print-data
         (format t "~%~%")
         (time:print-current-time)))
  )
)

;; Trains a net to recognize a pattern. Returns the plane/cell of the final
;  layer which responds most actively to the pattern.
(defun TRAIN (pattern-plane net
              &optional (iterations-per-layer-list '(4 4 4 1)))
  (let* ((layers-to-be-trained (cdr (net-layers net)))
        (num-layers (length layers-to-be-trained)))
    (cond ((not (= num-layers (length iterations-per-layer-list)))
           (ferror "~% Improper training iteration count list for
                training ~D layers: ~s"
                   num-layers iterations-per-layer-list))
          (t
           (setf (plane-cells (car (layer-planes (car (net-layers net)))))
                 pattern-plane)     ; assign input pattern
           (do* ((layer-list (cdr (net-layers net)) (cdr layer-list))
                                   ; loop over all layers
                 (layer (car layer-list) (car layer-list))
                 (iteration-list iterations-per-layer-list
                                  (cdr iteration-list))
                 (iterations (car iteration-list) (car iteration-list))
                                   ; # times to train layer
                 (result (caar (train-layer layer iterations))
                                   ; train each layer in succession
                         (caar (train-layer layer iterations))))
;                (result (caar (update-layer layer))     ; update the layer
;                        (caar (update-layer layer))))
                ((null (cdr layer-list)) result))
                                      ; return most active cell in final layer
           )
          )
    )
  )

; Trains a layer in a net to recognize a pattern. Continues training until
;  updating layer produces no more changes in the representative list
;  (returned by UPDATE-LAYER). At some point I'd like to remove the
;  ITERATIONS parameter and work only from changes in rep list, but
;  it is computationally prohibitive in the current version of the system.

(defun TRAIN-LAYER (layer &optional iterations)
  (do* ((old-reps nil new-reps)    ; record most active cell
        (new-reps (update-layer layer) (update-layer layer))
                                   ; re-adjust after training
        (i 0 (+ i 1)))
    ((or (equal old-reps new-reps) (>= i iterations)) new-reps)
                                   ; check if training complete
    (train-layer-aux layer new-reps))      ; perform training
  )

; Trains a layer in a net to recognize a pattern
(defun TRAIN-LAYER-AUX (layer &optional (representative-data-list
   (update-layer layer)))
    (let ((net-type (zl:lexpr-funcall #'extract-type-key
                                   (layer-local-parameters layer))))
; select appropriate training routine
      (zl:selectq  net-type
        (neocognitron
               (train-neocognitron-layer-aux
                  layer representative-data-list))
        (DIANN
               (train-DIANN-layer-aux
```

(continued on page 98)

4GL OR C

NOW THE CHOICE IS SIMPLE.

USE THE FAIRCOM® TOOLBOX AND GET BOTH 4GL SPEED AND C SOURCE CODE POWER.

Whether you need the development speed and convenience of 4GL programming or the low-overhead power capabilities of C source code, the FairCom ToolBox can meet the requirements of any professional developer!

INDUSTRIAL STRENGTH TOOLS

Develop applications the way you want with The ToolBox's industrial strength tools.

Development Environment by d-tree™

- Prototype generation
- Data dictionary
- Dynamic resource swapping
- Screen management
- Overlapped windows
- File restructuring
- Runtime portability
- Menu management

File Management by c-tree®

- Variable length records
- Key compression
- Client/Server architecture
- Ascending/Descending key segments
- Dynamic space reclamation
- Portable. Used in over 100 environments
- Variable length key fields
- High speed B+ trees

Report Generation by r-tree®

- Complex multi-line reports
- Multi-file access
- Complete layout control
- Conditional page breaks
- Nested headers and footers
- Unlimited control breaks
- Dynamic format specifications
- Horizontal repeats
- Powerful set functions

POWER AND FLEXIBILITY

Now you can create applications using the methods you like — whether it's 4GL convenience or C source code power. You can have it all with FairCom's introduction of The ToolBox Special Edition. And at $695 you get this power at a price you can afford.

ORDER TODAY

Order the FairCom Development ToolBox and use it for 30 days. No risk. If the FairCom ToolBox doesn't meet your development needs, just return the entire package for a full refund.

THE TOOLBOX

B Y • F A I R C O M •

CALL 1-800-234-8180 TODAY FOR YOUR FAIRCOM TOOLBOX

The ToolBox, Professional Edition $1,095.00
DOS, Unix, Xenix, VMS, OS2 Full source, single and multi-user support.

The ToolBox, Special Edition $695.00
Microsoft, Borland, Xenix, OS2 Object Libraries, single user only.

Upgrade to Professional Edition $400.00
Includes overnight delivery.

FAIRCOM® corporation
4006 West Broadway
Columbia, Missouri 65203
(314) 445-6833
FAX (3140 445-9698

CIRCLE NO. 128 ON READER SERVICE CARD

Listing Three *(Listing continued, text begins on page 28.)*

```
                layer representative-data-list))
      (ART2
            (train-ART2-layer-aux
                layer representative-data-list))
      (backpropagation
            (train-backprop-layer-aux
                layer representative-data-list))
                )
     )
   )

; Trains a layer in a neocognitron to recognize a pattern
(defun TRAIN-NEOCOGNITRON-LAYER-AUX (layer &optional (representative-data-list
 (update-layer layer)))
   (cond ((> *trace* 3)
        (format t "~%training layer.~%  Chosen Representative list:~%  ~s"
                representative-data-list)))
  (cond ((eq (layer-type layer) 'S)                        ; only train S-layers
        (mapcar
          #'(lambda (representative-data)
              (let* ((plane (car representative-data))
                     (pos (cadadr representative-data))
                     (prev-layer (layer-prev-layer layer))
                     (q-val (extract-layer-q layer))
                     (results
                       (do* ((all-connections (layer-i-connections layer)
                                               (cdr all-connections))
                          (connections (connections-for-pos layer plane pos))
                                                     ; extract connections
                             (all-i-vals-list (plane-i-weights plane)
                                               (cdr all-i-vals-list))
                                                  ; extract current weights
                             (old-i-vals (car all-i-vals-list)
                                          (car all-i-vals-list))
                             (all-i-vals-list (plane-i-weights plane)
                                               (cdr all-i-vals-list))
                             (all-i-vals (car all-i-vals-list)
                                          (car all-i-vals-list))
                             (all-i-weights-list (plane-i-weights plane)
                                                  (cdr all-i-weights-list))
                             (all-i-weights (car all-i-weights-list)
                                             (car all-i-weights-list))
                             (raw-result (train-plane prev-layer
                                                       connections old-i-vals
                                         all-i-vals all-i-weights q-val)
                                 ; perform training on excitatory connections
                               (train-plane prev-layer connections old-i-vals
                                             all-i-vals all-i-weights q-val))
                             (result raw-result
                                     (list (append (car result)
                                                   (car raw-result))
                                           (+ (cadr result)
                                              (cadr raw-result)))))
                                                          ; record result
                            ((null (cdr all-connections)) result)))
                                                          ; return result
                     (new-i-weight-list (car results))
                                                    ; adjust inhibitory weights
 (new-b-val (* q-val (compute-inhib-input (connections-for-pos layer plane pos)
                                          (c-weights-for-pos  plane pos)
                                                          prev-layer))))
       (setf (plane-i-weights plane)
                (transpose-on-type new-i-weight-list (layer-type layer)))
       (setf (plane-local-parameters plane)
        (insert-keyed-val (plane-local-parameters plane) '(:b ,new-b-val)))))
          representative-data-list))
       ((eq (layer-type layer) 'C)   representative-data-list)
                                     ; for C-layers, return representatives
       )
  )

;Trains a plane's CONNECTIONS to all planes in previous layer
(defun TRAIN-PLANE (prev-layer connections old-i-val-lists all-i-val-lists
                                                   all-i-weight-lists q-val)
 (do* ((connection-list connections (cdr connection-list))
                                                    ; extract connections
      (connection (car connection-list) (car connection-list))
      (c-val-list all-i-val-lists (cdr c-val-list)) ; extract current weights
      (c-vals (car c-val-list) (car c-val-list))
      (c-weight-list all-i-weight-lists (cdr c-weight-list))
      (c-weights (car c-weight-list) (car c-weight-list))
      (rev-old-i-vals (reverse old-i-val-lists))
      (old-i-vals (car old-i-val-lists)
                 (nth (- (length c-val-list) 1) rev-old-i-vals))
      (con-vals (train-plane-aux prev-layer connection old-i-vals
                     c-vals c-weights q-val)
                (train-plane-aux prev-layer connection old-i-vals
                     c-vals c-weights q-val))    ; perform actual training
      (new-i-weights (list (car con-vals))
                    (nconc new-i-weights
                           (list (car con-vals))))
      (vtotal (cadr con-vals) (+ vtotal (cadr con-vals))))
     ((null (cdr connection-list)) (list (list new-i-weights) vtotal)))
                                           ; return new connection weights
 )
```

End Listing Three

Listing Four

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: andy; Base: 10 -*-

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;       A Research Environment for the Instantiation of Neural Networks
;;;       Andrew  J. Czuchry, Jr. -- Georgia Institute of Technology
;;;       Georgia Tech Research Institute -- Artificial Intelligence Branch
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                       IDENTIFICATION functions
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Attempts to recognize a pattern using NET as a trained net
;  Returns the plane of the final layer which responds to the pattern.
(defun IDENTIFY (pattern-plane net)
  (setf (plane-cells (car (layer-planes (car (net-layers net)))))
       pattern-plane)                                  ; assign input pattern
  (let ((result (caar (last (update-net net)))))       ; update net
    (if (listp result)
        (car result))           ; return most active cell of final layer
    ))

; Updates entire net
;  Returns maximum value as nested set of lists
;   (((plane (value, pos)) ... (plane (value, pos)) layer1)
;   (((plane (value, pos)) ... (plane (value, pos)) layer2) ...)
(defun UPDATE-NET (net)
  (let ((  (extract-net-  net)))
    (do* ((layer-list  (cdr (net-layers net)) (cdr layer-list))
                                                       ; loop over all layers
          (layer (car layer-list) (car layer-list))
          (layer-max (update-layer layer) (update-layer layer))
                                                           ; update the layer
          (max (list (append layer-max (list layer)))
                             ; max value ((value, pos) plane)... layer) list
               (append max (list (append layer-max (list layer))))))
                                                          ; append each layer
         ((null (cdr layer-list)) max)
      )
    )
  )
```

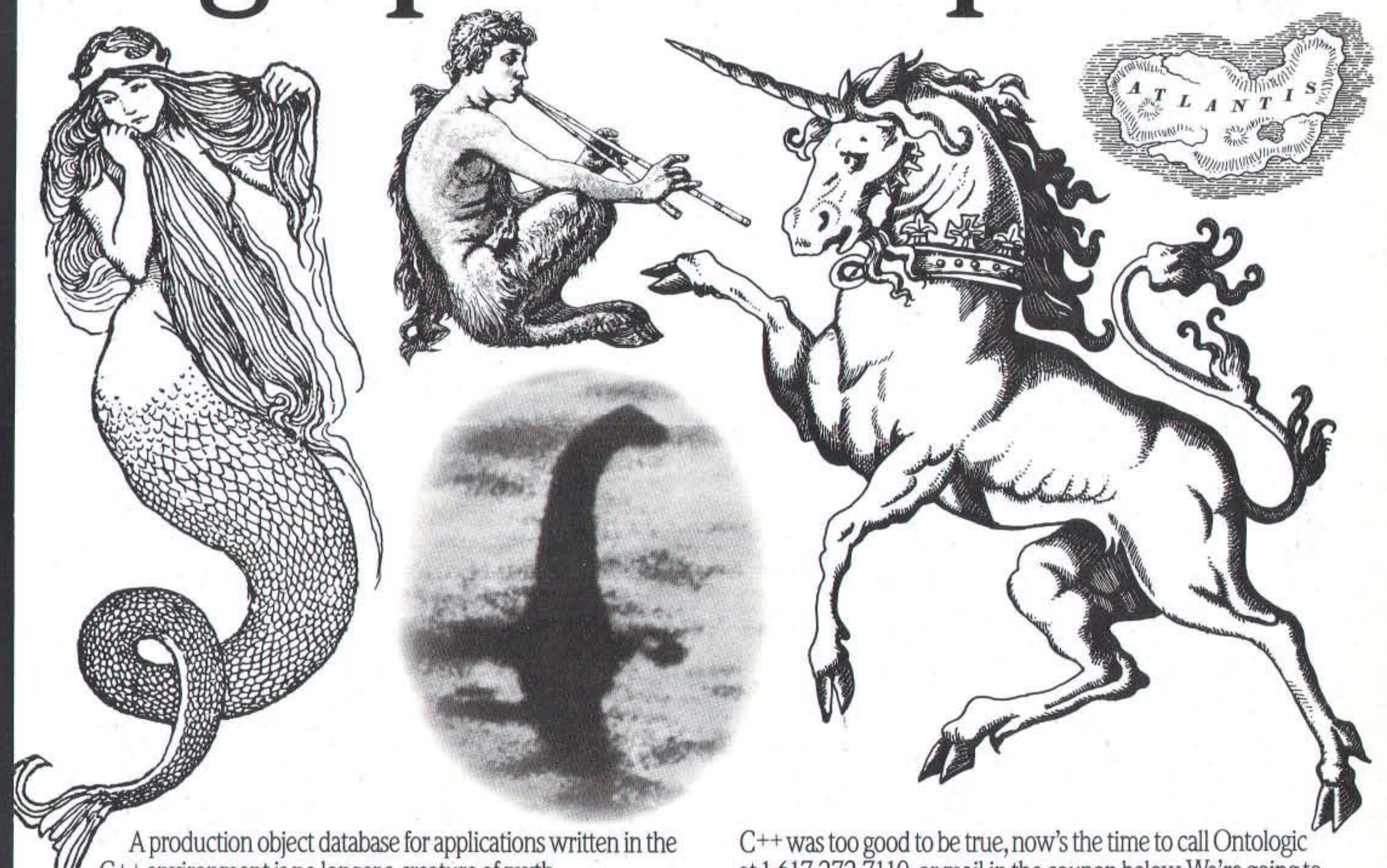
End Listing Four

Listing Five

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: andy; Base: 10 -*-

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;       A Research Environment for the Instantiation of Neural Networks
;;;       Andrew  J. Czuchry, Jr. -- Georgia Institute of Technology
;;;       Georgia Tech Research Institute -- Artificial Intelligence Branch
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;           Sample Variables of networks to be instantiated
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

(defvar *neocognitron-net*
       '(create-net
          :num-of-layers 7
          :num-of-planes-per-layer-list '(1 24 24 24 24 24 24)
          :plane-size-list '((16 16) (16 16) (10 10) (8 8) (6 6) (2 2) (1 1))
          :connection-pattern 'square
          :mask-size-list '((5 5) (5 5) (5 5) (5 5) (5 5) (2 2))
          :net-parameters '(:  0.5)
          :additional-parameter-list '(:net-type neocognitron
                                       :r-val-list (4.0 1.5 1.5)
                                       :q-val-list (1.0 16.0 16.0)
                                       :b-val-list (0.0 0.0 0.0)
                                       :orientation-list (8 1 8 1))
          )
  )
(defvar *neocognitron-net2*

       '(create-net
          :num-of-layers 7
          :num-of-planes-per-layer-list '(1 15 20 20 24 20 10)
          :plane-size-list '((24 24) (18 18) (12 12) (9 9) (8 8) (4 4) (1 1))
          :connection-pattern 'square
          :mask-size-list '((7 7) (3 3) (3 3) (4 4) (5 5) (4 4))
          :net-parameters '(:  0.5)
          :additional-parameter-list '(:net-type neocognitron
                                       :r-val-list (3.0 1.0 1.0)
                                       :q-val-list (10.0 18.0 18.0)
                                       :b-val-list (0.0 0.0 0.0)
                                       :orientation-list (12 1 12 1))
          )
       )
```

End Listings

Thanks to Ontologic, a production object database for C++ is no longer part of this picture.



A production object database for applications written in the C++ environment is no longer a creature of myth.

Introducing our Second Generation Object Database, ONTOS. The first and only real object database to support C++ that is deliverable and operational at customer sites today.

Fact is, because ONTOS is ideal for CASE, CAD and CAM applications, Index Technology, the country's leading CASE tool provider, recently signed a product agreement with Ontologic. As a result, Index Technology will base future product development on ONTOS.

This second generation object database fully supports and is 100% compatible with the C++ language. It is performance oriented, delivering 10-1,000 times the speed of a relational database system, making it competitive with custom-built databases. ONTOS's open architecture easily integrates with your existing development environment and supports industry standards including SQL.

If you thought a production object database for C++ was too good to be true, now's the time to call Ontologic at 1-617-272-7110, or mail in the coupon below. We're going to make a believer out of you.

I want the facts on ONTOS. DD4/90

☐ *Please have a salesperson call.*

☐ *Send me more information on your object database product right away.*

Mail coupon to: Ontologic, Inc., Three Burlington Woods, Burlington, MA 01803 (617) 272-7110

Name ______	*Title* ______
Company ______	*Phone* ( ) ______
Address ______	
City ______	*State* ______
Zip ______	
Current O/S ______	
Current Hardware ______	

Ontologic

Object Database Systems

CIRCLE NO. 41 ON READER SERVICE CARD

Listing One *(Text begins on page 46.)*

```
/****************************************************************************\
 *  tswit.c  -- iRMX II task switch time measurement.
 *  Compiler: iC-286 V4.1 (LARGE model). Q1 1989, by R. P. Kar
\****************************************************************************/

#include <stdio.h>
#include <rmxc.h>

#define MAX_LOOPS 500000L

unsigned        el_time, pri, status;
unsigned long   strt_sec, end_sec;
selector        task1_t, task2_t;
unsigned long   count1, count2;
float           ts_time;

/* "union" used to decompose a pointer into segment:offset */
typedef struct {unsigned offset; selector sel;} ptr_s;
union { unsigned *pointer; ptr_s ptr; } ptr_u;

void task1()
{
  for (count1 = 0; count1 < MAX_LOOPS; count1++)
    rqsleep(0, &status);                          /* Task switch happens here */
  rqdeletetask(NULL, &status);                                  /* delete self */
}

void task2()
{
  for (count2 = 0; count2 < MAX_LOOPS; count2++)
    rqsleep(0, &status);                         /* Task switch happens here */
  rqdeletetask(NULL, &status);                                 /* delete self */
}

/************************  MAIN PROGRAM  ************************/

main()
{

printf("\nTask Switch measurement\n   Each task runs %D times...\n\n",
           MAX_LOOPS);

/* Measure execution time of task1 and task2 when they are executed
   serially (without task switching). */
strt_sec = rqgettime(&status);                     /* Start of timing period */

  for (count1 = 0; count1 < MAX_LOOPS; count1++)
    /* rqsleep(0, &status) */ ;
  for (count2 = 0; count2 < MAX_LOOPS; count2++)
    /* rqsleep(0, &status) */ ;
end_sec = rqgettime(&status);                        /* End of timing period */

el_time = (unsigned)(end_sec - strt_sec);

/* Place a pointer to any variable in union "ptr_u", so the data segment
   of this program becomes known. */
ptr_u.pointer = &status;

/* Get main program's priority level */
pri = rqgetpriority (NULL, &status);

/* Create two (identical) tasks, which just switch between themselves */
task1_t = rqcreatetask (pri+1, task1, ptr_u.ptr.sel, 0L, 512, 0, &status);
if (status != 0) printf("rqcreatetask error\n");

task2_t = rqcreatetask (pri+1, task2, ptr_u.ptr.sel, 0L, 512, 0, &status);
strt_sec = rqgettime(&status);                    /* Start of timing period */

/* Set main program's priority below task 1,2 so they run to completion */
rqsetpriority( (selector)0, pri+2, &status );
rqsleep( 0, &status );

end_sec = rqgettime(&status);                       /* End of timing period */

/* Set main program back to initial priority */
rqsetpriority( (selector)0, pri, &status );

el_time = (unsigned)(end_sec - strt_sec) - el_time;
ts_time = ( (float)el_time * 1000000.0 ) / ((float)MAX_LOOPS * 2.0) ;
printf("   Task switch time = %5.1f microseconds\n", ts_time);
dqexit(0);
}
```

End Listing One

Listing Two

```
/****************************************************************************\
 *  preempt.c  -- iRMX II preemption time benchmark.
 *  Measures the time for 1 preemptive task switch + 1 non-preemptive task
 *  switch. Compiler: iC-286 V4.1 (LARGE model). Q4 1989, by R. P. Kar
\****************************************************************************/

#include <stdio.h>
#include <rmxc.h>

/* NOTE: 100,000 iterations takes about 35 minutes on a 16 MHz 386 PC */
#define MAX_LOOPS 100000L

/* Note: This is a CPU-dependent value. It must be set such that
 * the execution time for this loop: for (j=0; j < ONE_TICK; j++) spare++
 * is slightly longer than one iRMX sleep period.
*/
#define ONE_TICK 4200

unsigned        pri, status, i, spare, el_time;
unsigned long   strt_sec, end_sec;
selector        task1_t, task2_t, co_conn;
unsigned long   count1, count2;
float           preempt_time;

/* "union" used to decompose a pointer into segment:offset */
typedef struct {unsigned offset; selector sel;} ptr_s;
union { unsigned *pointer; ptr_s ptr; } ptr_u;

/* The lower priority task. It sits in delay loop waiting to be preempted. */
void task1()
{
  unsigned loc_status;
  for (count1 = 0; count1 < MAX_LOOPS; count1++)
    for (i = 0; i < ONE_TICK; i++) ++spare;                /* Waste time */
  printf("deleting task 1\n\n");
  rqdeletetask(NULL, &loc_status);                          /* delete self */
}

/* The higher priority task. When it goes to sleep (once in every loop) iRMX
 * makes a non-preemptive switch to the other task; when the sleep period ends
 * this task preempts the other task.
 */
void task2()
{
  unsigned loc_status;
  for (count2 = 0; count2 < MAX_LOOPS; count2++)
    /* When rqsleep is called, task switch to lower priority task happens.
     * When 1 clock period is over, other task is preempted and control
     * returns to the next line.
     */
    rqsleep(1, &loc_status);
  printf("\ndeleting task 2\n");
  rqdeletetask(NULL, &loc_status);                    /* delete self */
}

/************************  MAIN PROGRAM  ************************/
main()
{

printf("\nPreemption time benchmark\n   Each task runs %D times...\n\n",
           MAX_LOOPS);

/* Measure execution time of task1 and task2 when they are executed
 * serially (without task switching or preemption).
 */
strt_sec = rqgettime(&status);                     /* Start of timing period */
  for (count1 = 0; count1 < MAX_LOOPS; count1++)
    for (i = 0; i < ONE_TICK; i++) ++spare;
  for (count2 = 0; count2 < MAX_LOOPS; count2++)
end_sec = rqgettime(&status);                      /* End of timing period */

el_time = (unsigned)(end_sec - strt_sec);

printf("    Execution without premption & task switching took %u seconds\n",
             el_time);

/* Place a pointer to any variable in union "ptr_u", so the data segment
   of this program becomes known.
 */
ptr_u.pointer = &status;

/* Get main program's priority */
pri = rqgetpriority (NULL, &status);

task1_t = rqcreatetask (pri+2, task1, ptr_u.ptr.sel, 0L, 512, 0, &status);
if (status != 0) printf("rqcreatetask error\n");

task2_t = rqcreatetask (pri+1, task2, ptr_u.ptr.sel, 0L, 512, 0, &status);

strt_sec = rqgettime(&status);                    /* Start of timing period */

/* Set main program's priority below task 1,2 so they run to completion */
rqsetpriority( (selector)0, pri+3, &status );
rqsleep( 0, &status );

end_sec = rqgettime(&status);                     /* End of timing period */

/* Set main program back to initial priority */
rqsetpriority( (selector)0, pri, &status );
el_time = (unsigned)(end_sec - strt_sec) - el_time;
preempt_time = ( (float)el_time / (float)MAX_LOOPS ) * 1000000.0;
printf("     Preemption time + task switch time = %5.1f microseconds\n",
                   preempt_time);
dqexit(0);
}
```

End Listing Two

Listing Three

```
/****************************************************************************\
 *  ltncy.c -- iRMX II interrupt latency benchmarking program.
 *  Method:  This program first sets up an interrupt handler for an unused
 *     interrupt level. It then reads the count in the system timer (timer
 *     0 on the 8254 chip) and simulates an external interrupt to the CPU by
 *     a cause$interrupt instruction. The interrupt handler latches timer 0,
 *     so this program can read it again after the handler returns control.
 *     The difference in the two timer-count values is the interrupt latency.
 *                    Oct 1989, by R. P. Kar
\****************************************************************************/

#include <stdio.h>
#include <rmxc.h>

/* Define base address of 8254 (Programmable Interval Timer) chip */
#define         PIT_ADDR   0x40

unsigned        status, ticks, timer_cnt1, timer_cnt2;
unsigned        dummy_w;
unsigned char  pri, lo_cnt1, hi_cnt1, lo_cnt2;

extern void int_hndlr();

/**************************   MAIN PROGRAM   **************************/
main ()
```

```
{
printf("                        ***  WARNING ***\n\n");
printf("  This program assumes that timer and interrupt controller\n");
printf("  hardware is fully compatible with the IBM PC/AT\n\n");

/* Set up local handler for IRQ3 on master 8259 */
rqsetinterrupt( 0x38, 0, int_hndlr, (selector)0, &status );
disable();                                        /* Disable interrupts */

/* Latch and read timer 0 value. Interrupt handler will latch it again */
outbyte( PIT_ADDR + 3, 0 );

/* The following two instructions read the value latched in counter 0. They
    are unavoidable measurement overhead and inflate the interrupt latency
    by a few clock cycles.
 */
lo_cnt1 = inbyte( PIT_ADDR );
hi_cnt1 = inbyte( PIT_ADDR );

/* Activate the interrupt handler. It will latch timer 0 and return. */
causeinterrupt(59);

/* The interrupt handler has latched the timer 0 count. Now read it. */
lo_cnt2 = inbyte( PIT_ADDR );
dummy_w = (unsigned int)inbyte( PIT_ADDR );
timer_cnt2 = (unsigned int)lo_cnt2 + (dummy_w << 8);

enable();                                      /* Re-enable interrupts */

dummy_w = (unsigned int)hi_cnt1;
timer_cnt1 = (unsigned int)lo_cnt1 + (dummy_w << 8);

/* Calculate difference in timer counts (timer counts DOWN to 0) */
if (timer_cnt1 > timer_cnt2)
    ticks = timer_cnt1 - timer_cnt2;
else                             /* Rare case when timer has wrapped around */
     ticks = timer_cnt1 + (0xffff - timer_cnt2 + 1);

/* Display results */
printf("   Interrupt latency = %u timer ticks\n", ticks);

/* Note that timer is pulsed by 1.19 MHz crystal */
printf("                     = %4.1f microseconds\n\n", ((float)ticks)/1.19 );

rqresetinterrupt( 0x38, &status );
dqexit(0);
}
```

End Listing Three

Listing Four

```
; latch.asm -- Interrupt handler. Merely latches timer 0 in a
; PC/AT (or hardware compatible computer).

      NAME latch

latch SEGMENT PUBLIC

int_hndlr PROC FAR
PUBLIC int_hndlr
  PUSHA
  XOR  AX,AX
  OUT  43H, AL     ; Latch 8254 counter 0
  POPA
  IRET
int_hndlr ENDP
latch ENDS
      END
```

End Listing Four

Listing Five

```
/*************************************************************************\
 *  semshuf.c -- iRMX II semaphore shuffle measurement.
 *  Measures the latency (within iRMX) for a task to acquire
 *  a sempahore that is owned by another equal-priority task.
 *  Compiler: Intel iC-286 V4.1 (LARGE model). Q3 1989, by R. P. Kar
\*************************************************************************/

#include <stdio.h>
#include <rmxc.h>

#define MAX_LOOPS 100000L

enum      YESNO {NO, YES} sem_exch;
unsigned        el_time, status;
selector        task1_t, task2_t, sem_t;
unsigned char   pri;
unsigned long   count1, count2, maxloop2;
unsigned long   strt_sec, end_sec;
float           semshuf_time;

/* "union" used to decompose a pointer into segment:offset */
typedef struct {unsigned offset; selector sel;} ptr_s;
union { unsigned *pointer; ptr_s ptr; } ptr_u;

void task1()
{
  unsigned rem_units, t1_status;
  for (count1 = 0; count1 < MAX_LOOPS; count1++)
  {
    /* Task waits here until other task relinquishes semaphore */
    if (sem_exch == YES)
      rem_units = rqreceiveunits( sem_t, 1, 0xffff, &t1_status );
    rqsleep(0, &t1_status);
    if (sem_exch == YES)
      rqsendunits( sem_t, 1, &t1_status );
    rqsleep(0, &t1_status);
```

(continued on page 102)

Windows? OS/2 PM?

Save 8 Hours Of Repetitive Grunt Work Every Time You Write A Prototype Or Application.

Winpro/2 And Winpro/PM: The Fastest, Easiest Way For C Programmers To Write Windows And OS/2 PM Applications.

Once you've created a resource file that describes your application's menus, accelerator keys, and dialog box layouts, it will ordinarily take you more than eight hours of repetetive, tedious, error-prone programming to create a working program skeleton that incorporates the resources you've defined.

*Now you can go from a resource file to a working prototype in just five minutes, with **Winpro/2** or **Winpro/PM**.* What's more, you'll have readable, well-commented C language source code that's more extensive, more complete, and more suitable for further development than the code other products produce.

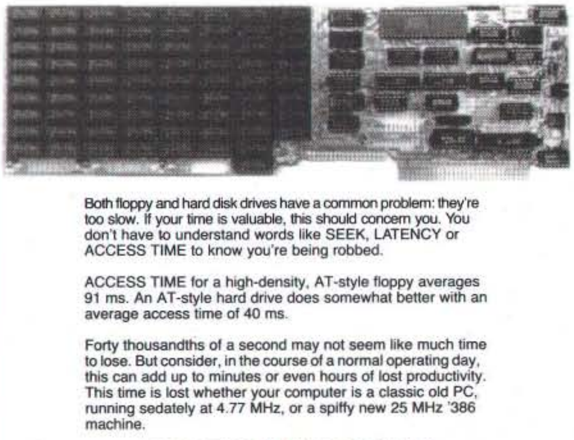
FREE TRIAL OFFER: Call us at 201-447-3270 and let us send you a copy of **Winpro/2** or **Winpro/PM**. Try it for 30 days. If it is what we promise--a great time-saver and a valuable programming tool--*then* mail us a check for $395. If for any reason you *don't* want to keep the product, just send it back to us *and you don't owe us a cent.*

Xian Corporation
625 North Monroe St., Ridgewood, NJ 07450

Call: (201) 447-3270
Fax: (201) 447-2547

CIRCLE NO. 256 ON READER SERVICE CARD

STUCK IN THE SLOW LANE?



Both floppy and hard disk drives have a common problem: they're too slow. If your time is valuable, this should concern you. You don't have to understand words like SEEK, LATENCY or ACCESS TIME to know you're being robbed.

ACCESS TIME for a high-density, AT-style floppy averages 91 ms. An AT-style hard drive does somewhat better with an average access time of 40 ms.

Forty thousandths of a second may not seem like much time to lose. But consider, in the course of a normal operating day, this can add up to minutes or even hours of lost productivity. This time is lost whether your computer is a classic old PC, running sedately at 4.77 MHz, or a spiffy new 25 MHz '386 machine.

COMPUTING IN THE FAST LANE!

SemiDisk is a dedicated disk emulator. A SemiDisk looks to your computer like an additional disk drive. A drive with a big difference: It's extremely FAST! Average access times are less than 0.01 ms. Our Blue Flame II models are available in sizes from 2 to 8 megabytes per single-slot card. Multiple cards can be used together to form logical drives of up to 32 megabytes each. Add our Battery Backup Unit and you'll have a non-volatile drive that can't be beat. Call or write for benchmark data compiled with PC Magazine's Benchmark tests.

Blue Flame II prices start below $500. Blue Flame II cards with capacities under 8 megabytes are field upgradable in 1 megabyte steps. The Battery Backup option costs $130. Call for information on SemiDisk products for S-100 and Epson QX-10/QX-16.

SemiDisk Systems, Inc.
P.O. Box GG
Beaverton, Oregon 97075
phone: [503] 626-3104
FAX: [503] 643-0625

CIRCLE NO. 212 ON READER SERVICE CARD

AMX™

REAL-TIME MULTITASKING KERNEL

8086/88, 80x86/88 80386 *68000/10/20 Z80, 64180, 8080/85*

- Fast, reliable operation
- Compact and ROMable
- PC peripheral support
- DOS file access
- C language support
- Preemptive scheduler
- Time slicing available
- Configuration Builder
- Complete documentation
- Intertask messages
- Message exchanges
- Dynamic operations
  - — task create/delete
  - — task priorities
  - — memory allocation
- Event Manager
- Semaphore Manager
- List Manager

No Royalties
Source Code Included

Manual only	**$75 US**
AMX 86	**$3000 US**

(Shipping/handling extra)

Call for prices for other processors.

KADAK Products Ltd.
206-1847 West Broadway
Vancouver, B.C., Canada
V6J 1Y5
Telephone: (604) 734-2796
Fax: (604) 734-8114

CIRCLE NO. 259 ON READER SERVICE CARD

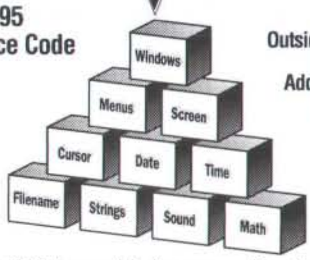
ATTENTION ASSEMBLY LANGUAGE PROGRAMMERS

BUILDING BLOCKS FOR THE ASSEMBLY LANGUAGE PROGRAMMER

QUANTASM POWER LIB

Price $99.95
With Source Code $299.95

S/H $5.00
Outside U.S. $15.00
CA Residence
Add 7% Sales Tax
MC and VISA



- ► *Over 256 Assembly Language Routines*
- ► *No Need to Re-Invent the Wheel Every Time You Write a New Application*
- ► *Permits Faster Application Development*
- ► *Overlapping Windowing System in Less Than 3K*
- ► *Fast and Compact*
- ► *Ideal For Programming TSR's*
- ► *C and Pascal Interfaces Available*
- ► *MASM 5.X Compatible*

QUANTASM CORPORATION
19855 Srevens Creek Blvd, Suite 154
Cupertino, CA 95014
(408) 244-6826

ALSO AVAILABLE:
ASMFLOW $99.95
An Assembly Language Flow Charting and Source Code Analysis Tool

CIRCLE NO. 326 ON READER SERVICE CARD

Listing Five *(Listing continued, text begins on page 46.)*

```
  }
  rqdeletetask( (selector)0, &t1_status );                   /* delete self */
}

void task2()
{
  unsigned rem_units, t2_status;
  for (count2 = 0; count2 < MAX_LOOPS; count2++)
  {
    /* Task waits here until other task relinquishes semaphore */
    if (sem_exch == YES)
      rem_units = rqreceiveunits( sem_t, 1, 0xffff, &t2_status);
    rqsleep(0, &t2_status);
    if (sem_exch == YES)
      rqsendunits( sem_t, 1, &t2_status );
    rqsleep(0, &t2_status);
  }
  rqdeletetask( (selector)0, &t2_status );                  /* delete self */
}

/************************  MAIN PROGRAM  ************************/
main()
{

printf("\nSemaphore shuffle benchmark\n   %U shuffles...\n\n", MAX_LOOPS*2);

/* Get priority of main program */
pri = rqgetpriority( (selector)0, &status );

/* Create 2 tasks; measure their execution time WITHOUT semaphore shuffling */
 sem_exch = NO;

task1_t = rqcreatetask( pri+1, task1, ptr_u.ptr.sel, 0L, 512, 0, &status );
if (status != 0) printf("Create task error\n");

task2_t = rqcreatetask( pri+1, task2, ptr_u.ptr.sel, 0L, 512, 0, &status );

strt_sec = rqgettime(&status);                  /* Start of timing period */

/* Set main program's priority below task 1,2 so they run to completion */
rqsetpriority( (selector)0, pri+2, &status );
rqsleep( 0, &status );

end_sec = rqgettime(&status);                   /* End of timing period */

el_time = (unsigned)(end_sec - strt_sec);
printf("    Execution time without semaphore shuffle = %u secs\n",el_time);

/* Set main() back to original priority level */
rqsetpriority( (selector)0, pri, &status );

sem_t = rqcreatesemaphore( 1, 1, 0, &status );
if (status != 0) printf("Create sem error\n");

/* Re-create 2 tasks. This time they will shuffle semaphore between them. */
sem_exch = YES;

task1_t = rqcreatetask( pri+1, task1, ptr_u.ptr.sel, 0L, 512, 0, &status );
if (status != 0) printf("Create task error\n");

task2_t = rqcreatetask( pri+1, task2, ptr_u.ptr.sel, 0L, 512, 0, &status );

strt_sec = rqgettime(&status);                     /* Start of timing period */

/* Set main program's priority below task 1,2 so they run to completion */
rqsetpriority( (selector)0, pri+2, &status);
rqsleep( 0, &status );

end_sec = rqgettime(&status);                      /* End of timing period */
el_time = (unsigned)(end_sec - strt_sec) - el_time;
printf("    %U semaphore exchanges took %u seconds\n", MAX_LOOPS*2, el_time)

semshuf_time = ( (float)el_time / ((float)MAX_LOOPS * 2.0) ) * 1000000.0;
printf("    ..... %5.1f microseconds per shuffle\n\n", semshuf_time);

dqexit(0);
}
```

End Listing Five

Listing Six

```
/***************************************************************************\
 *  deadbrk.c -- iRMX II Deadlock break-time measurement.
 *  A low, medium and high priority task is created. Deadlock occurs
 *  when the following chronological sequence happens:
 *     (1) low priority task takes exclusive control of a critical resource
 *     (2) medium or high priority task preempts it.
 *     (3) high priority task requests resource; gets suspended
 *     (4) Medium priority task runs, blocking other two tasks indefinitely
 *  This situation is handled in iRMX by acquiring a "region" before
 *  using critical resource, and relinquishing it after use. This benchmark
 *  measures the overhead involved in "breaking the deadlock".
 *  Compiler: Intel iC-286 V4.1 (LARGE model). Q3 1989, by R. P. Kar
\***************************************************************************/

#include <stdio.h>
#include <rmxc.h>

#define MAX_LOOPS 10000
/* Note: This is a CPU-dependent value. It must be set such that the
 * execution time for this loop: for (j=0; j < DELAY; j++) spare++
 * is slightly longer than one iRMX sleep period.
 */
#define DELAY     4000

unsigned        el_time, spare, status;
selector        task1_t, task2_t, task3_t, region_t;
unsigned char   pri;
enum            YESNO {NO, YES} dead_brk;
```

```c
unsigned long  count1, count2, count3, max_loops;
unsigned long  strt_sec, end_sec;
float          deadbrk_time;

/* "union" used to decompose a pointer into segment:offset */
typedef struct {unsigned offset; selector sel;} ptr_s;
union { unsigned *pointer; ptr_s ptr; } ptr_u;

/* Low priority task */
void task1()
{
  unsigned t1_status, j;
  while (1)
  {
    if (count1 == max_loops)
    { printf("deleting task1\n");
      rqdeletetask( (selector)0, &t1_status );           /* delete self */
    }
    /* Get control over critical region */
    rqreceivecontrol( region_t, &t1_status );

    for (j = 0; j < DELAY; j++) spare++;                  /* delay loop */

    count1++;
    rqsendcontrol( &t1_status );
  }
}

/* Medium priority task. Only uses CPU time and sleep periodically. */
void task2()
{
  unsigned j, t2_status;

  while (1)
  {
    if (count2 == max_loops)
    { printf("deleting task2\n");
      rqdeletetask( (selector)0, &t2_status );           /* delete self */
    }
    for (j = 0; j < DELAY/4; j++) spare++;                /* delay loop */

    rqsleep(1, &t2_status);
    count2++;
  }
}

/* High priority task. Potential deadlock when it tries to gain control
   of the "region" resource, because low-priority task holds region mostly.
 */
void task3()
{
  unsigned t3_status;

  while (1)
  {
    if (count3 == max_loops)
    { printf("deleting task3\n");
      rqdeletetask( (selector)0, &t3_status );           /* delete self */
    }
    rqsleep(1, &t3_status);
    /* Ask for control of the region. Relinquish control immediately after
       receiving it. If task1 is not already holding region, this should
       take very little time. Otherwise, OS must break deadlock.
     */
    if (dead_brk == YES)
    { rqreceivecontrol( region_t, &t3_status );
      rqsendcontrol( &t3_status );
    }

    count3++;
  }
}

/********************** Main program ***********************/
main( argc, argv )

unsigned argc;
char *argv[];

{
if (argc > 1)
  max_loops = (unsigned)atoi(argv[1]);
else max_loops = MAX_LOOPS;

printf("\nDeadlock break time benchmark\n   %U loops...\n\n",max_loops);

/* Get priority of main program */
pri = rqgetpriority( (selector)0, &status );

/* Create three tasks. Task1 has lowest priority, task3 has highest.
 * Measure their execution time WITHOUT deadlocks.
 */
count1 = count2 = count3 = 0;
dead_brk = NO;

task1_t = rqcreatetask( pri+3, task1, ptr_u.ptr.sel, 0L, 512, 0, &status );
if (status != 0) printf("Create task error\n");

task2_t = rqcreatetask( pri+2, task2, ptr_u.ptr.sel, 0L, 512, 0, &status );

task3_t = rqcreatetask( pri+1, task3, ptr_u.ptr.sel, 0L, 512, 0, &status );

strt_sec = rqgettime(&status);                     /* Start of timing period */

/* Set main program's priority below task 1,2,3 so they run to completion */
rqsetpriority( (selector)0, pri+4, &status );
while ( (count1 < max_loops) || (count2 < max_loops) || (count3 < max_loops) )
  rqsleep( 10, &status );
```

(continued on page 104)

INTUITIVE.

MetaWINDOW™ is a complete graphic application development environment that lets you tap into your creativity, *instead* of trying your patience. *MetaWINDOW* gives you over *200* powerful functions that let you work ***faster*** and ***easier*** than ever before with support to over *15* popular C and Pascal compilers!

MetaWINDOW provides industry standard PostScript and X-Window style drawing attributes and a QuickDraw superset that makes it quick and easy to port your existing PC & Mac applications. And with dynamic runtime support for over *100* popular graphics cards, you won't be wasting time with bothersome configuration sessions! *MetaWINDOW* also provides the ultimate in fast graphics and text drawing functions with extensive rasterOp flexibility that makes rubberbanding or dragging complex images a snap! All this, plus full, illustrated documentation, free expert technical support, and ***royalty-free*** licensing. *Call now to receive your **free working demo** or download direct from our BBS at (408) 438-5368!*

MetaWINDOW	***$250***
TurboWINDOW/C *(or Pascal)*	***$125***
QuickWINDOW/C	***$125***
FontWindow	***$125***

30-DAY MONEY BACK GUARANTEE.

ORDER HOTLINE:
800-332-1550

CIRCLE NO. 156 ON READER SERVICE CARD

TM

METAGRAPHICS
SOFTWARE CORPORATION
269 MOUNT HERMON ROAD, SCOTTS VALLEY, CA, 95066
IN CA:(408)438-1550 FAX:(408)438-5379 BBS:(408)438-5368 (1200/2400)

Listing Six *(Listing continued, text begins on page 46.)*

```
end_sec = rqgettime(&status);                    /* End of timing period */

el_time = (unsigned)(end_sec - strt_sec);
printf("    Execution time without deadlocks = %u secs\n\n",el_time);

/* Set main() back to original priority level */
rqsetpriority( (selector)0, pri, &status );

/* Create a "region". To ensure mutually exclusive access to a critical
   resource a task must acquire the region first */
region_t = rqcreateregion( 1, &status );
if (status != 0) printf("Create region error\n");

count1 = count2 = count3 = 0;
dead_brk = YES;

/* Re-create tasks 1,2,3. Now tasks 1 & 3 will compete for region */
task1_t = rqcreatetask( pri+3, task1, ptr_u.ptr.sel, 0L, 512, 0, &status );
if (status != 0) printf("Create task error\n");

task2_t = rqcreatetask( pri+2, task2, ptr_u.ptr.sel, 0L, 512, 0, &status );

task3_t = rqcreatetask( pri+1, task3, ptr_u.ptr.sel, 0L, 512, 0, &status );

strt_sec = rqgettime(&status);                    /* Start of timing period */

/* Set main program's priority below tasks 1,2,3 so they run to completion */
rqsetpriority( (selector)0, pri+4, &status);
while ( (count1 < max_loops) ¦¦ (count2 < max_loops) ¦¦ (count3 < max_loops) )
  rqsleep( 10, &status );

end_sec = rqgettime(&status);                    /* End of timing period */

el_time = (unsigned)(end_sec - strt_sec) - el_time;
printf("    %U deadlock resolutions took %u seconds\n", count3, el_time);

deadbrk_time = ( (float)el_time/(float)count3 ) * 1000000.0;
printf("    ..... %6.1f microseconds per resolution\n\n", deadbrk_time);

dqexit(0);
}
```

End Listing Six

Listing Seven

```
/************************************************************************\
 *  it_msg.c -- iRMX II inter-task data message latency measurement.
 *  First run the code of two tasks serially (no messages sent). Then
 *  create two tasks and a "mailbox" and measure how much extra time is
 *  needed to send a fixed number of messages from task 1 to task 2.
 *      Compiler: iC-286 V4.1 (LARGE model). Q4 1989, by R. P. Kar
\************************************************************************/

#include <stdio.h>
#include <rmxc.h>

#define MAX_LOOPS 200000L

unsigned long  strt_sec, end_sec;
selector       task1_t, task2_t, mbox_t;
unsigned       pri, el_time, msg_length, status;
unsigned long  count1, count2;
float          it_msg_time;
char           msg_buf[10] = "MESSAGE\0",
               recv_buf[];

/* "union" used to decompose a pointer into segment:offset */
typedef struct {unsigned offset; selector sel;} ptr_s;
union { unsigned *pointer; ptr_s ptr; } ptr_u;

/* This task sends data messages, to task 2 that is waiting to receive */
void task1()
{
  unsigned loc_status;

  for (count1 = 0; count1 < MAX_LOOPS; count1++)
  {  /* Put a serial # on the message */
     msg_buf[8] = (unsigned char)count1 / 256;
     rqsenddata( mbox_t, msg_buf, 10, &loc_status );
  }
  printf("Task 1 exiting....\n");
  rqdeletetask(NULL, &status);                          /* delete self */
}

/* This task receives the data messages */
void task2()
{
  unsigned loc_status;

  for (count2 = 0; count2 < MAX_LOOPS; count2++)
    msg_length = rqreceivedata( mbox_t, recv_buf, 0xffff, &loc_status );
  printf("    Last message received... %s %u (length %u)\n", recv_buf,
                           (unsigned)recv_buf[8], msg_length );
  rqdeletetask(NULL, &status);                         /* delete self */
}

/***************************  MAIN PROGRAM  ***************************/
/* First parameter to "rqcreatemailbox" ==> data mailbox, FIFO queues */
#define MBOX_FLAG 0x0020

main()
{
printf("   Inter-task message latency measurement\n");
printf("      Sending %D data messages...\n\n", MAX_LOOPS);

/* Set up a mailbox for inter-task data communication */
mbox_t = rqcreatemailbox( MBOX_FLAG, &status );
if (status != 0) printf("rqcreatemailbox error\n");

/* Measure serial execution time of tasks 1,2 (without messages) */

strt_sec = rqgettime(&status);                    /* Start of timing period */
  for (count1 = 0; count1 < MAX_LOOPS; count1++)
  {  /* Put a serial # on the message */
     msg_buf[8] = (unsigned char)count1 / 256;
     /* rqsenddata( mbox_t, msg_buf, 10, &loc_status ); */
  }
  for (count2 = 0; count2 < MAX_LOOPS; count2++)
    /* msg_length = rqreceivedata( mbox_t, recv_buf, 0xffff, &loc_status ) */;
end_sec = rqgettime(&status);                      /* End of timing period */

el_time = (unsigned)(end_sec - strt_sec);

/* Place a pointer to any variable in union "ptr_u", so the data segment
   of this program becomes known.
 */
ptr_u.pointer = &status;

/* Get main program's priority level */
pri = rqgetpriority (NULL, &status);

task1_t = rqcreatetask (pri+2, task1, ptr_u.ptr.sel, 0L, 512, 0, &status);
if (status != 0) printf("rqcreatetask error\n");

/* Task 2 is created with a higher priority than task 1. This ensures that if
 * it is waiting at a mailbox for a message from task 1, it will be scheduled
 * as soon as the message is sent.
 */
task2_t = rqcreatetask (pri+1, task2, ptr_u.ptr.sel, 0L, 512, 0, &status);

strt_sec = rqgettime(&status);                      /* Start of timing period */

/* Set main program's priority below task 1,2 so they run to completion */
rqsetpriority( (selector)0, pri+3, &status );
rqsleep( 0, &status );

end_sec = rqgettime(&status);                      /* End of timing period */

/* Set main program back to initial priority */
rqsetpriority( (selector)0, pri, &status );

el_time = (unsigned)(end_sec - strt_sec) - el_time;
it_msg_time =  ( (float)el_time * 1000000.0 ) / (float)MAX_LOOPS ;
printf("    Inter-task message latency + task switch time = %6.1f
microsecs\n",
         it_msg_time);

/* Delete mailbox */
rqdeletemailbox( mbox_t, &status );

dqexit(0);
}
```

End Listings

QNX* Users!

Call for catalogue of QNX and QNX support software & hardware

- **All Quantum Products**
- **Dynaterm Terminals**
- **Ethernet: TCP/IP for QNX**
- **LispC**
- **Compilers: C, C86, FTN77+, BASIC**
- **Menu Packages**
- **Development tools including CShell**
- **QFAX, BTREE II, Bernoulli Drivers**
- **QNX terminal emulation for DOS**
- **ScreenBlox-ReportBlox**

*QNX is a registered trademark of Quantum Software Systems, Ltd.

T&T Computer Products, Inc.
QNX Specialists
P.O. Box 14010
Tulsa, OK 74159 U.S.A.
918/663-1879   FAX 918/663-4054

MC/VISA

CIRCLE NO. 219 ON READER SERVICE CARD

I can write a 4.5 gigabyte OLTP database application, with 130 different record types, that gives up to 172 simultaneous DOS LAN users 1-2 second response time. All in less than 640K.

This person is a (check one):

- ☐ *Dreamer*
- ☐ *Shameless Liar*
- ☐ *MDBS IV Developer*

The answer is "MDBS IV Developer." Because the database application he's talking about is not only possible, but real. It belongs to the U.S. Department of Energy's Fermi National Accelerator Laboratory.

Fermi Lab needed an on-line materials handling system to track the millions of items used by the D.O.E. complex, and MDBS IV was the only database that allowed them to develop a full-scale production application using an IBM PC-based LAN.

No relational database on the market could do the job. And even with substantial hardware upgrades, none could approach the speed of MDBS IV.

McDonnell Douglas, NASA, Ernst & Young, and many others agree. They've found that MDBS IV is the only way they can get real mainframe performance on existing hardware under 640K. What's more, MDBS IV applications are easily moved from one environment to another: DOS, OS/2, UNIX and VMS. And, of course, MDBS IV speaks SQL.

In every major industry, successful application developers are switching to MDBS IV because it lets MIS retain mainframe-style control of the network without a mainframe-size investment. Applications are easier to write on PCs, too, and this helps them stay on top of their applications backlog.

Call **1-800-323-3629** for a free technical white paper on MDBS IV, plus case histories that relate to your industry. International inquiries, call (317) 447-1122. Because, for today's increasingly complex database needs, MDBS IV is the only right answer.

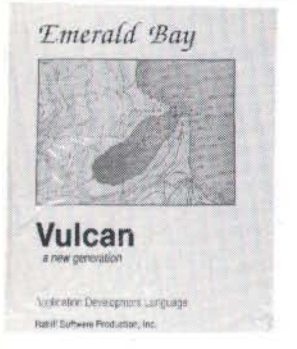
mdbs®

MDBS IV • Object/1 • KnowledgeMan/2 • GURU
1834 Walden Office Square, Suite 250, Schaumburg, IL 60173
(800) 323-3629/(708) 303-6300/FAX (708) 303-6830

VF91383A/po#1152
Dillon & Partners/2-14-90
great faces/339-2933/kristin

VF91383B/po#1152
Dillon & Partners/2-16-90
great faces/339-2933/caroline

Over 2,500 Software Tools


Emerald Bay
Vulcan
a new generation
Application Development Language
Ratliff Software Production, Inc.

Emerald Bay Vulcan

by Ratliff Software

Emerald Bay's Vulcan is a next generation dBASE. Both a DBMS language, interpreter and compiler, Vulcan includes user-defined functions, nested reads, mouse-support, save and restore screens, and multi-dimensional arrays. If you know the dBase language, you know Vulcan, whose programs are smaller and eaiser to maintain. The compiler is in the performance class of Clipper, Foxbase, and Quicksilver.

List: $395 Ours: $329

FastFaxts 1747-002

386 DEVELOPMENT TOOLS

	List	Ours
386 Max	75	69
DESQview 386	220	189
FoxBASE+/386	595	479
High C - 386	895	875
Hoops exted. DOS MetaWare	795	759
VM/386	245	229

AI EXPERT SYSTEMS

Exsys Professional	795	695
Logic Gem by Sterling Castle	99	89
VP Expert	249	219

AI-LANGUAGES

ARITY Combination Package	1095	989
PC Scheme LISP- by TI	95	85
TURBO PROLOG V2.0	150	119

ASSEMBLERS

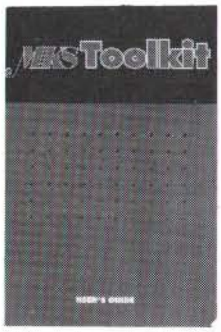
386 IASM/LINK by Pharlap	495	Call
MS Macro Asm	150	105
Turbo Assembler/Debugger	150	109

C LANGUAGE COMPILERS

High C-286 - by MetaWare	595	590
High C-386 - by MetaWare	895	875
Instant C/16M	795	769
Lattice C - 6.0	250	189
Microsoft C5.1 - w/CodeView	450	299
Micorsoft QuickC	99	69
w/serial mouse	199	137
Turbo C - by Borland	150	Call
WATCOM C7.0	395	329
WATCOM C7.0 386	895	799
Zortech C	Call	Call

CASE & PROTOTYPERS

Dan Bricklin Demo II	195	185
Interactive EasyFlow	150	135
Instant Replay - Nostradamus	150	139
Matrix Layout - flow chart	200	179
MetaDesign by Meta Software	350	339
Pro-C	495	429
ProtoFinish by Genesis	300	279
R&R Code Generator	149	129
Show Partner F/X	395	349



MKS Toolkit

by MKS

Provides over 150 UNIX utilities which can be linked together through pipes and I/O redirection to perform complex tasks. Includes: MKS Korn Shell, Awk programming and prototyping language, Vi text editor plus, sort merge, profiler and menu driven installation and configuration programs. A 250 page reference manual, and 260 pages of tutorials provide an excellent UNIX learning environment. DOS & OS/2.

FastFaxts 469-004

Visible Computer 80286	100	85

BASIC & ADD-ONS

BAS-C Commercial	495	439
dB/Lib	139	125
MS QuickBASIC V4.5	99	69
ProBas Lib by Hammerly	159	Call
ProRef by Hammerly	50	Call
ProScreen by Hammerly	99	Call
ProMath by Hammerly	99	95
QBase	149	139
QuickPak Professional	149	139
Softcode/w Template	100	89

BOOKS/VIDEOS

Mouse Programmer's Reference	30	24
Programming Windows	25	19
Running MS-DOS, 4th Edition	23	18
Zortech C Video Course	300	Call
Zortech C++ Video Course	Call	Call

Visable Analyst	595	585

COBOL

MS COBOL V3.0	900	Call
Realia COBOL	995	859

COMMUNICATIONS ADD-ONS

C Asynch Lib	249	219
C Asynch Manager 3.0	189	155
Essential Lib by S. Mountain	329	309
Greenleaf Comm Library	299	249
QuickComm	139	129

DATABASE MANAGEMENT

Cause by Maxem	295	259
CLARION	845	549
(Professional Developer V2.1)		
dbFast Windows	395	329
D the data language	395	359
Magic PC	499	479
PC Plus -Texas Instruments	2950	1999
Paradox V3.0	725	545
R-BASE for DOS	725	499
Tas Lan Support	200	185
Tas Professional	499	459

DBASE

Clipper Summer '87 MU	695	459
db/Fast	99	89
dBASE IV	795	Call
dbFAST Windows	395	329
dbFAST/PLUS	249	219
dBMAN V	295	275
Force by Sophco	695	579
FoxPro	795	495
FoxBASE + - V2.1	395	279
QuickSilver Diamond	599	399
Sunshow Professional	100	95

DBASE ADD-ONS

AdComm 99	129	119
Artful Lib	200	Call
CodeBASE 4	295	279
dGE - business graphics	195	179

Knowledge Pro's Toolkit	129	119
R&R Relational Reportwriter	149	139
Scrimmage by Synergy - screen/menu	99	89
Sycero db - single user	495	479
UI Programmer - Dev's Version 2.0	595	479

DEBUGGERS/ DISASSEMBLERS

Dis Doc	130	119
Periscope II & III - breakout switch	Varies	Call
RE:Source by Genesoft	150	119
SoftProbe	395	345
Sourcer	100	89
sPORT by WHS	80	Call
Trapper	200	189
Zortech C Debugger	90	Call
Zortech C++ Debugger	150	Call

DEVELOPMENT TOOLS

AsmFlow	100	89

The BEST linker for MS-DOS.....

.RTLink/Plus *by Pocket Soft*

The only MS-DOS linker with CodeView support for multiple/nested overlays. Advanced technology allows more overlays, overlay areas, symbols, objects, classes, and sections than any other linker. Smart caching of most recently accessed overlay sections in conventional, expanded, and extended memory (New XMS specifications).
Easy to use Profiler provides detailed performance statistics without source code modification. Run-Time Libraries (RTLs) speed development time and reduce disk-space/modem-time requirements. Source code provided for overlay manager and Profiler's read-analysis utility. LIBLIST provides detailed information on inter-relationships between modules-great for optimizing and designing overlay schemes. Highest score of all linkers. Reviewed by ***PC Week*** 1/29/90.

LIst: $495 Ours: $419 ***FastFaxts 1277-003***

dQuery MU	495	469
FLIPPER Graphics Library	195	179
FUNCcy.LIB	195	179
Pro Clip	149	125
R&R Relational Reportwriter & Clipper module	165	149
SilverComm Library	189	169
SilverPaint	100	89
Steve Straley's Toolkit	180	149
Tom Rettig's Library	100	85

DBASE TOOLS

Buzzwords dAnalyst	295	269
CLEAR + for dBASE	200	179
dAnalyst Plus	195	189
Dbase Online - 6 pop-up refs.	149	129
BRIEF w/dBRIEF	285	Call
dBug - source debugger	195	179
dBX - dBASE III to C	550	529
dSalvage	100	89
Genifer - code generator	395	269

Autoflow C/Pascal	299	279
C Programmers Toolbox	99	89
CLEAR + for C	200	169
Codan	395	349
dAnalyst for Clipper	295	269
Documenter	295	245
Inside	125	119
Knowledge Pro	495	419
Knowledge Pro Windows	695	589
MKS Lex & Yacc	249	199
MKS RCS	189	175
Poly Doc-SU	199	179
PC-Lint	139	115
Plink 86 Plus - overlays	495	399
PolyMake	149	135
PVCS Personal	149	135
PVCS Professional	695	659
.RTLINK - by Pocket Soft	295	279
.RTLINK Plus	495	419
Sbrowse	245	Call
Source Print by Powerline	99	Call

WATCOM C7.0

by WATCOM

WATCOM C7.0 is a complete C development system. Offering significant performance in the speed and size of compiled programs. This package includes: ANSI C Optimizing Compiler and run-time library, graphics library, windowed source-level debugger, development tools, and express C integrated environment. Important features include: tight code generated by the optimizing compiler, loop optimizations for even faster code, remote debugging capability for debugging large programs; and compilation and optimization of large files.
MS-DOS

List: $395 Ours:$329

FastFaxts 1044-002

THE PROGRAMMER'S

Call today for complete technical information and advice, our free catalog, literature and *solid value*. ***Prices subject to change without notic***

For Professional Developers!

Sourcer w/BIOS Pre-ProC	140	129
TLIB	100	89
Zortech C++ Tools	150	139

EDITORS

BRIEF	199	Call
Edix	195	179
Epsilon - like EMACS	195	169
KEDIT - like XEDIT. V/4	150	139
SPF/PC - V2.1	245	199
Vedit Plus	185	139

FILE ADD-ONS

Btrieve ISAM - V5.0	245	199
Btrieve/N - multiuser	595	479
Xtrieve + - by Novell	595	479
C-Data Manager	150	139
c-tree by Faircom - source	395	329
d-tree - data dictionary	495	459
r-tree - report generator	295	245
c-tree - w/r-tree	650	549

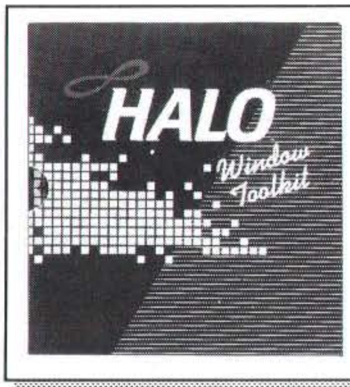
Greenleaf SuperFunctions	299	239
Turbo C Tools by Blaise	149	134

GRAPHIC ADD-ONS

Codemaster II	289	269
Essential Graphics	399	349
GraphiC	395	319
Greenleaf Makeform-DOS	125	119
GSS Dev't Toolkit	595	525
Halo '88 - 140 + devices	395	279
Hoops	795	759
MetaWINDOW	250	209
MetaWINDOW/PLUS	325	289
PCX Programmer's Toolkit	195	179
Pizazz Plus	149	129
QuickWindows Advanced	139	129
Sunscan	250	229
Sunshow Adv Image Toolkit	250	239
Sunshow C Image	250	229

NETWORKS

Netback	245	229

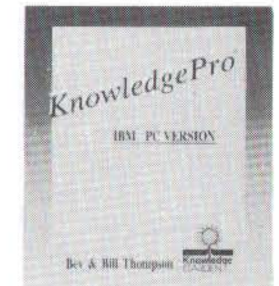


HALO Window Toolkit
by Media Cybernetics
The Windowing Alternative

HALO Window Toolkit is a graphical user interface tool that speeds development of graphics and imaging applications.
- Sophisticated memory manager detects and uses extended, expanded, or disk memory.
- Includes HALO graphics toolkit.
- Supports variety of graphics displays: high resolution, imaging devices, printers and scanners.
- Practical time-saver for MS C programmers.
- Source code compatible development path to target DOS and OS/2 operating environments.

List $595 Ours: $519

FastFaxts 86-024



KnowledgePro
by Knowledge Garden

Introducing the **DOOR** into **WINDOWS.** KnowledgePro is a high level object-oriented language for Microsoft Windows or DOS. Integration of OOP, Hypermedia and Expert Systems technology lets you create applications and intelligent documents quickly. Supports DDE & DLL. No runtime fees for applications

List: $695 (Windows) Ours: Call
List: $495 (DOS) Ours: Call
Graphics and Database Toolkits $129 each.
Call 518-766-3000 with questions, or

FastFaxts 1419-003

OS/286 by Eclipse	495	459
OS/386 by Eclipse	495	459

OTHER LANGUAGES

Modula-2 Dev. System- by Logitech	249	229
RPG II Dev. Systems	1600	1469
TopSpeed MODULA-2 Compiler	100	89
Turbo Prolog V2.0	150	119

OTHER PRODUCTS

386 Max Professional	130	119
Carbon Copy Plus	199	159
CO/SESSION	249	229
Dan Bricklin's PageGarden	100	89
Duplicator Toolkit	80	75
Fastback Plus	189	159
File Shuttle by GETC	120	109
Flow Charting II Plus	229	179
FM-Plus	100	89
Headroom by Helix	130	89
HiJaak	149	139

TURBO PASCAL

graphics-Menu	149	139
Say What	50	Call
Topaz	75	69
TP2C	249	229
Turbo Analyst	99	89
Turbo ASYNCH PLUS	189	169
Turbo Pascal 5.0 by Borland	150	109
Turbo POWER TOOLS PLUS	149	109
Turbo Professional	125	109
Turbo Programmer	300	269

TEXT SCREEN ADD-ONS

AEWindows	225	209
C Windows Toolkit	100	89
C Worthy w/forms	295	Call
Greenleaf DataWindows	395	339
Hi-Screen XL	149	129
Hi-Screen XL Professional	325	289
Hyperinterface- TFN, C, PAS menus	100	95

CBTREE - Source	195	179
CQL - full w/ source	395	359
dBC ISAM III - by Lattice	250	229
db_FILE - network model	Varies	Call
db_RETRIEVE - SQL	Varies	Call
XQL	795	649
WKS Library	195	189

FORTRAN

FOR_C - FORTRAN '77 w/so	875	789
Fortran Opt. Compiler	450	309
Lahey FORTRAN F77L	595	549
Lahey Personal FORTRAN	95	Call
MS FORTRAN - CodeView	450	309
Spindrift - DOS, windows	149	129

GENERAL ADD-ONS

C + O Class Library	195	179
C Tools Plus - V6.0	149	109
C Utilities-by South Mountain	185	137
Greenleaf Functions	209	189

NETmanager	1495	1295
cc:Mail 25 user	695	649
DaVinci eMAIL	995	895
Netware SQL	595	519
Netware C Interface	295	239

OBJECT-ORIENTED

Actor	695	639
C Talk	150	139
C Talk/views	450	419
C++ 80386 by Intek	495	469
Guidelines C++ V2.0	395	379
Smalltalk/V	100	85
Zortech C ++	200	Call

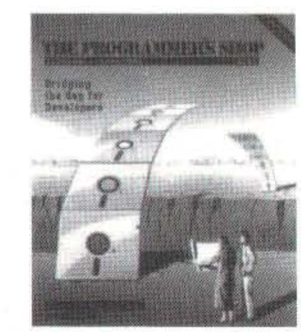
OS - SUPPORT

DESQview	130	109
MKS Toolkit - Unix shell	249	229
MS Windows/286	99	69
MS Windows/386	195	139
Development Toolkit/Wind	500	319
Omni View	80	69

CLARION
PROFESSIONAL DEVELOPER®
by Clarion

Professional Developer 2.1 is an immensely powerful, easy-to-use DBMS application developer, capable of cutting applications development time by 50%. Allows import/export of dBASE BASIC and DIF files; interfaces with routines from C and Assembler. Applications generator for creating screens, menus, windows, reports, and tables. Includes Report Writer for creation of ad hoc reports and queries. Comes with built-in LAN support, with no run-time system required for distribution. Recent winner of PCWeek poll of corporations using programmable databases.

List: $845 Ours: $549 ***FastFaxts 1005-004***



FastFaxts service!

Instant product literature via fax 24 hours a day!
Call 617-740-0025, from your fax machine, and enter FastFaxts code. (See Catalog)

Free Catalog!

Over 1,750 development tools listed. Call for your free copy.
1-800-421-8006

Mention "DD490"

Hyperpad by Brightbill	149	129
LapLink III	150	129
Link & Locate ++ - ROM MSC	395	349
Math Advantage	495	475
Mirror II	70	59
Net Lib	249	Call
Norton Commander	149	119
Norton Utilities Advanced	150	109
Paginate by Accumatics	100	89
PC Anywhere III	145	129
PC Tools Deluxe	129	109
PC Kwik Power Pack	130	119
QEMM-386	100	89
Remote2	195	139
Resident C by South Mountain	99	85
w/source	249	199
Service Diagnostic Kit	195	Call
SMX (College)	95	Call
SpinRite II	89	Call
Time $heet	150	135
Coterm 220	245	219

JAM by JYACC	595	469
MEWEL	179	149
POWERSCREEN by Blaise	149	109
Vitamin C - source, menus	225	169
VC Screen - painter	149	139
Vermont Views-Obj+Source	890	819
Zortech Pro Screen	70	Call

UNIX/XENIX

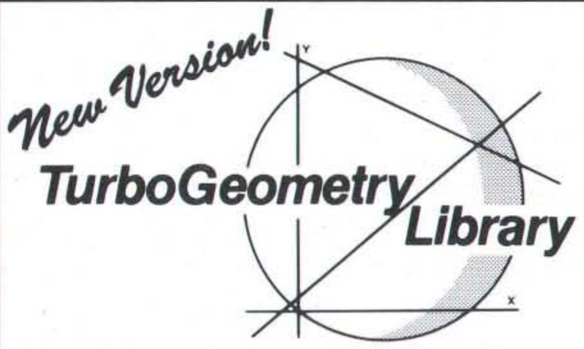
Application Devel. 2 user	1445	1349
Edix - editor	425	399
ESIX/V 386 Devlp (2 user)	595	569
Guidelines C++ for V/386 V2.0	495	479
Informix SQL	995	919
JAM JYACC Applic. Mgr.	many	Call
NDP Fortran-386	795	749
Network Developer 2 user	1495	1399
Vermont Views	1595	1349
Workstation Devel. 2 user	1995	1869
Workstation Starter 2 user	495	469
XENIX 286 Complete System	1495	1279
XENIX 386 Complete System	1595	1369

Credit Card orders processed only when product is shipped.

SHOP • 800-421-8006

5 Pond Park Road, Hingham, MA 02043 • **Canada 800-446-3846** • MA 617-740-2510 • FAX: 617-749-2018

CIRCLE NO. 197 ON READER SERVICE CARD



A Library of 2D & 3D Geometric Routines

New 3.0 upgrade! Twice as many 2D and 3D geometric routines (over 300). Now with: Surfacing • Solids • Spherical Geometry • 3D Clipping • Tangents. Also includes additional HideLine algorithms • Intersections • Volumes • Areas • Transforms • Perspectives • Decomposition • and more. Easy to use manual. 30 day guarantee. $199.95 with source, S&H incl. Foreign $225.00, S&H incl. TurboGeometry Library 3.0 is available in Turbo Pascal, Quick Pascal, MSC, Turbo C, Power C, Zortech C. Needs MS/PC DOS, IBM/PC or Compatibles. We accept VISA, MasterCard, PO, Chk. USA negotiable funds only.

Disk Software, Inc.

2116 E. Arapaho Rd.
Suite 487
Richardson, TX 75081
USA

214-423-7288
214-423-7288 FAX
800-635-7760

CIRCLE NO. 117 ON READER SERVICE CARD

Listing One *(Text begins on page 56.)*

```
/* FONT1 STRUCTURE DEFINITIONS */

struct fontlhead {      /* standard character font header */
   unsigned char fnttype;        /* font structure type: */
        /* non-compressed type = 0x16 or compressed type = 0x14 */
   char fntname[13];   /* font name: always followed with a '.set' extension*/
   unsigned char fntcheck; /* check digit: verifies a Data Transforms font: */
        /* non-compressed font = 0xba, compressed font = 0xdc */
   unsigned char fntbase;
                   /* baseline count (in pixels) from top to bottom, top = 0 */
   unsigned char fnttotal; /* total characters in font:*/
        /* limited to the lower 94 ASCII characters: 0x21 - 0x7E */
   unsigned char fntstart;       /* starting character */
   unsigned char fntstatus; /* proportional or non-proportional: 0=non-prop.*/
   unsigned char fnthsize;      /* horizontal cell size in pixels */
   unsigned char fntvsize;      /* vertical cell size in pixels */
   unsigned char fntbytes;      /* number of horizontal bytes in current cell*/
   unsigned char fntspaceh;     /* space bar horizontal size in pixels */
   unsigned char fntchargap;    /* pixels between characters default */
   unsigned char fntlfgap;      /* pixels between linefeeds default */
   int fntlength;               /* total length of file */
   unsigned char fntpitch;      /* italics pitch (0 = none) */
             /* bits 0 - 6 ... number of scanlines to skip */
             /* bit  7 ... 0 = decrement xpos, 1 = increment xpos */
   unsigned char fntinvert;      /* 0 = dont invert, 1 = invert */
   unsigned char fnthbold;       /* number of overlapping bits horizontal */
   unsigned char fntvbold;       /* number of overlapping bits vertical */
   unsigned char fnthmag;        /* integral horizontal bit magnification */
   unsigned char fntvmag;        /* integral vertical bit magnification */
   unsigned char fnthfract;      /* fractional horizontal bit magnification */
   unsigned char fntvfract;      /* fractional vertical bit magnification */
   unsigned char fntdirection;
                    /* Print direction 0=left to right,1..3=counterclock */
   unsigned char fntrot90;       /* rotation 0=up, 1..3=counterclock 1...3 */
   unsigned char fnthflip;       /* horizontal flip 0 = no, 1 = yes */
   unsigned char fntvflip;       /* vertical flip 0 = no, 1 = yes */
   unsigned char fntcolor;       /* color of font */
           /* bits 0 - 3 ... foreground color */
           /* bit 0 ... strike black ribbon */
           /* bit 1 ... strike blue ribbon */
           /* bit 2 ... strike red ribbon */
           /* bit 3 ... strike yellow ribbon */
           /* bits 4 - 7 ... background color */
           /* bit 4 ... strike black ribbon */
           /* bit 5 ... strike blue ribbon */
           /* bit 6 ... strike red ribbon */
           /* bit 7 ... strike yellow ribbon */
   unsigned char fntsubtype;     /* subcategory type of this font */
           /* 0 = normal font subtype */
           /* 1 = equation roman font subtype */
           /* 2 = equation symbol font subtype */
   unsigned char fntunused[18]; /* unused bytes */
};

struct font1 {  /* standard character font (type = 0x16) */
   struct fontlhead fhd;         /* font header */
   char *fntcellptr[FONT1TOTAL + 1];
                     /* offsets from beginning of file to character bitmaps */
   char fntcellwidth[FONT1TOTAL + 1];   /* cell widths if proportional */
};
```

End Listing One

Listing Two

```
/* FONT2 STRUCTURE DEFINITIONS */

struct font2 {
   struct fontlhead fhd2;                     /* font header */
   unsigned int fnt2cellseg[FONT2TOTAL + 1];
                                  /* array: segment pointers to characters*/
   int fnt2cellhsize[FONT2TOTAL + 1];   /* array: cell horiz sizes in bits */
   int fnt2cellhoffset[FONT2TOTAL + 1]; /* array: cell horiz offsets in bits*/
   int fnt2cellhbytes[FONT2TOTAL + 1];  /* array: cell horiz size in bytes */
   int fnt2cellvsize[FONT2TOTAL + 1];   /* array: cell vert sizes in bits */
   int fnt2cellvoffset[FONT2TOTAL + 1]; /* array: cell vert offsets in bits */
};
```

End Listings

You'll never use malloc again!

VENN DYNAMIC SET MANAGER

Powerful Storage Management
- Unlimited Size Arrays/Structures (Data Sets)
- Transparent Migration: Expanded Memory/Disk
- Automatic Storage Size Adjustment
- Transient or Permanent Data Sets

Direct Storage References
- Local Windows into Any Region of Data Set
- All Standard C Operations on Local Windows

Intelligent Operators
- Compare and Copy
- Any Set, Any Size, Any Storage Combination

Relational Style Operators
- Restrict, Intersect, Sort, Delta, and more
- Find Matches
- Create Subsets
- Sort Data Sets
- Perform Updates

Easy To Use
- Simple Declarations
- Sample Applications
- One Linkable Library
- Microsoft C and Turbo C Versions

ONLY $129.95

VENN SET PROCESSING SOFTWARE
1-800-628-8369 P.O. BOX 1008
VISA/MC ANN ARBOR, MI 48106-1008

CIRCLE NO. 414 ON READER SERVICE CARD

Listing One *(Text begins on page 72.)*

```c
/*****************************************************************************
File:    Kaboooom.c
Purpose: Allows the user to 'walk' through a minefield; a detector shows
  how many mines are immediately adjacent to you. As you visit a cell, it
  leaves a marker telling you how many were next to you, and you have the
  ability to mark cells with a character (assumably to mark mines).
 Changes:
 11/10/89 (tdeii) If you call the program with "/s" or "/S", it gives
  you a "safer" game, where it does not let you walk on spaces that you
  have marked (whether there is a mine there or not!)
  11/11/89 (tdeii) Allows you to press "?" and get some help starting at your
  current position; will mark mines that it knows (by deducing their position),
  and "visit" places that it knows are safe. This propagates until it cannot
  deduce anything else (see EvaluatePosition).
*****************************************************************************/
#include "stdio.h"
#include "stdarg.h"
#include "stdlib.h"
#include "dos.h"
#include "conio.h"
#include "string.h"

#define SCREEN_X 80
#define SCREEN_Y 25
#define GRID_X 15
#define GRID_Y  9
#define TRUE 1
#define FALSE 0

#define bEMPTY    0
#define bVISITED  1
#define bBOMB     2
#define bCURRENT  3
#define bFINISH   4
#define bEXPLODED 5

#define MAKECOLOR(fore,back) ((back)*16+(fore))

typedef int BOOL;
typedef struct tagADJACENCYGROUP {
  int BombCount;          /* Number of bombs located in this adj. group */
  int CellCount;          /* Number of cells filled                     */
  int Cell[8][2];         /* x,y coordinates of up to 8 cells           */
} ADJACENCYGROUP;

int  Board[GRID_X][GRID_Y];                /* Board; see codes above (bXXX)    */
int  UserMark[GRID_X][GRID_Y];             /* User marks; 0 = none, 'M' = mine */
int  nNumMines;                            /* Number of mines on board         */
int  UserX, UserY;                         /* Current user X and Y position    */
BOOL bShowBombs;                           /* TRUE if program shows bombs (it  */
                                           /* does this after you win or lose) */
BOOL bSafeGame;                            /* TRUE if program does not let you */
                                           /* walk on mines you have marked    */
ADJACENCYGROUP AdjacencyGroup[GRID_X][GRID_Y]; /* AG for each board pos        */
char szClear[79] = "
                   ";
void Pause(void)
{
  if (getch() == 0) getch();
}

void GetXY_(int *pX, int *pY)
{
  union REGS regs;
  regs.h.ah = 3;
  regs.h.bh = 0;                                            /* display page 0 */
  int86(0x10, &regs, &regs);
  if (pY != NULL) {
    (*pY) = regs.h.dh;
  }
  if (pX != NULL) {
    (*pX) = regs.h.dl;
  }
}
void GotoXY_(int x, int y)
{
  union REGS regs;
  regs.h.ah = 2;
  regs.h.dh = (unsigned char)y;
  regs.h.dl = (unsigned char)x;
  regs.h.bh = 0;                                            /* display page 0 */
  int86(0x10, &regs, &regs);
}
int nRandom(int nMax)
{
  return ((int)((double)rand() / RAND_MAX * (double)nMax));
}
void DisplayChar(int x, int y, char cChar, int nColor)
{
  char far *cPos;
  if ((x>=0 && x<SCREEN_X) &&
      (y>=0 && y<SCREEN_Y)) {
    cPos = MK_FP( 0x0b800,((x + y*80) << 1));
    *cPos = cChar;
    *(cPos+1) = (char)nColor;
  }
}
int CountMines(int x, int y)
{
  int i, j;
  int nCount;
  nCount = 0;
  for (i=-1; i<=1; i++) {
    for (j=-1; j<=1; j++) {
      if ((x+i >= 0) && (x+i < GRID_X) &&
          (y+j >= 0) && (y+j < GRID_Y)) {
        if ((Board[x+i][y+j] == bBOMB) ||
            (Board[x+i][y+j] == bEXPLODED)) {
          nCount++;
        }
      }
    }
  }
  return (nCount);
}
void DisplayCell(int x, int y)
{
  int Char;
  Char = UserMark[x][y];
  if (Char == 0) {
    Char = 32;
  }
  DisplayChar(x*4+1, y*2+1, Char, MAKECOLOR(14,1));
  DisplayChar(x*4+3, y*2+1, Char, MAKECOLOR(14,1));
  switch (Board[x][y]) {
    case bEMPTY:                                        /** Empty cell   **/
      Char = ' ';
      break;
    case bVISITED:                                      /** Visited cell **/
      Char = '0' + CountMines(x, y);
      break;
    case bBOMB:                                         /** Bomb cell!   **/
      if (bShowBombs) {
        Char = 15;
      } else {
        Char =' ';
      }
      break;
    case bCURRENT:                                      /** Current pos  **/
      Char = 2;
      break;
    case bFINISH:                                       /** Finish cell  **/
      Char = 19;
      break;
    case bEXPLODED:                                     /** Exploded!    **/
      Char = 15;
      break;
  }
  if (Char != 0) {
    DisplayChar(x*4+2, y*2+1, Char, MAKECOLOR(14,1));
  }
}
void Initialize(void)
{
  unsigned int     nRand;       /* seed for random number generator        */
  struct dostime_t sDosTime;    /* time structure; used for above seed     */
  _dos_gettime(&sDosTime);
  nRand = (unsigned int)((sDosTime.hsecond * 600) +
                         (sDosTime.second * 10) +
                         (sDosTime.minute / 6));
  srand(nRand);
}
void PaintBoard(void)
{
  int x, y, i;
  for (x=0; x<SCREEN_X; x++) {
    for (y=0; y<SCREEN_Y; y++) {
      DisplayChar(x, y, ' ', MAKECOLOR(14, 1));
    }
  }
  /** Draw left and right sides **/
  DisplayChar(0, 0, 218, MAKECOLOR(14,1));  /*upper left corner */
  DisplayChar(GRID_X*4, 0, 191, MAKECOLOR(14,1)); /*upper right corner */
  for (y=1; y<=GRID_Y; y++) {
    DisplayChar(0, y*2, 195, MAKECOLOR(14,1));  /* left edge */
    DisplayChar(GRID_X*4, y*2, 180, MAKECOLOR(14,1)); /* right edge */
  }
  DisplayChar(0, GRID_Y*2, 192, MAKECOLOR(14,1));  /* lower left corner */
  DisplayChar(GRID_X*4, GRID_Y*2, 217, MAKECOLOR(14,1)); /*lower right */
  /** Draw inside corners **/
  for (x=1; x<GRID_X; x++) {
    DisplayChar(x*4, 0, 194, MAKECOLOR(14,1)); /* top edge */
    for (y=1; y<GRID_Y; y++) {
      DisplayChar(x*4, y*2, 197, MAKECOLOR(14,1)); /* intersections */
    }
    DisplayChar(x*4, GRID_Y*2, 193, MAKECOLOR(14,1)); /* bottom edge */
  }
  /** Draw connecting lines **/
  for (x=0; x<=GRID_X; x++) {
    for (y=0; y<=GRID_Y; y++) {
      if (y != GRID_Y) {
        DisplayChar(x*4, y*2+1, 179, MAKECOLOR(14,1)); /* verticals  */
      }
      if (x != GRID_X) {
        for (i=1; i<4; i++) {
          DisplayChar(x*4+i, y*2, 196 , MAKECOLOR(14,1)); /* horizontals */
        }
      }
    }
  }
  GotoXY_(0, SCREEN_Y - 1);
  for (x=0; x<GRID_X; x++) {
    for (y=0; y<GRID_Y; y++) {
      DisplayCell(x, y);
    }
  }
}
void SetUpBoard(void)
{
  int i, j;
  int nMines;
  BOOL bDone;
  char cBuffer[80];
  bShowBombs = FALSE;
  /** First, get number of bombs **/
  nNumMines = 0;
  while ((nNumMines < 10) || (nNumMines > 40)) {
    GotoXY_(0, 24);
```

(continued on page 110)

Listing One *(Listing continued, text begins on page 72.)*

```
    printf("How many bombs do you want? (10-40)?? ");
    fgets(cBuffer, sizeof(cBuffer), stdin);
    sscanf(cBuffer, "%d", &nNumMines);
  }
  /** next, clear out board & user scratchpad **/
  for (i=0; i<GRID_X; i++) {
    for (j=0; j<GRID_Y; j++) {
      Board[i][j] = 0;
      UserMark[i][j] = 0;
    }
  }
  for (nMines=0; nMines<nNumMines; nMines++) {
    bDone = FALSE;
    while (!bDone) {
      i = nRandom(GRID_X);          /* First you roll it, */
      j = nRandom(GRID_Y);          /* Then you pat it,   */
      if ((Board[i][j] == bEMPTY) &&
          (!((i <= 1) && (j <= 1))) &&
          (!((i >= GRID_X - 2) && (j >= GRID_Y - 2)))
        ) {
        bDone = TRUE;
      }
    }
    Board[i][j] = bBOMB;            /* Then you mark it with a 'B' */
  }
  /* Set user at position 0, 0 */
  UserX = 0;
  UserY = 0;
  Board[0][0] = bCURRENT;
  /* Set finish (hq) at position GRID_X, GRID_Y */
  Board[GRID_X - 1][GRID_Y - 1] = bFINISH;
  /* Display board on screen */
  PaintBoard();
}
BOOL Travel(int dx, int dy)
{
  int NewX, NewY;                /* New X and Y coordinates of user      */
  BOOL bInvalid;                 /* TRUE if trying to walk off board     */
  BOOL bAbort;                   /* TRUE if user won or lost (abort game) */
  BOOL bBombWalk;                /* TRUE if user tried to walk on a bomb  */

  bAbort = FALSE;
  NewX = UserX + dx;
  NewY = UserY + dy;
  bInvalid = FALSE;
  bBombWalk = FALSE;
  if ((NewX < 0) || (NewX >= GRID_X)) {
    bInvalid = TRUE;
  }
  if ((NewY < 0) || (NewY >= GRID_Y)) {
    bInvalid = TRUE;
  }
  if ((!bInvalid) && (bSafeGame) && (UserMark[NewX][NewY] == 'M')) {
    bInvalid = TRUE;
    bBombWalk = TRUE;
  }
  if (bInvalid) {
    GotoXY_(0, SCREEN_Y - 1);
    printf("** INVALID MOVE ** ... press any key...");
    if (bBombWalk) {
      printf("(You must un-mark it.)");
    }
    Pause();
    GotoXY_(0, SCREEN_Y - 1);
    printf(szClear);
  } else {
    if (Board[NewX][NewY] == bBOMB) {
      bAbort = TRUE;
      Board[UserX][UserY] = bVISITED;
      DisplayCell(UserX, UserY);
      Board[NewX][NewY] = bEXPLODED;
      DisplayCell(NewX, NewY);
      GotoXY_(0, 22);
      printf("******** YOU HAVE STEPPED ON A BOMB!! ********");
      Pause();
      GotoXY_(0, 22);
      printf(szClear);
      GotoXY_(0, 22);
    } else {
      if ((NewX == GRID_X-1) && (NewY == GRID_Y-1)) {
        bAbort = TRUE;
        Board[UserX][UserY] = bVISITED;
        DisplayCell(UserX, UserY);
        Board[NewX][NewY] = bCURRENT;
        DisplayCell(NewX, NewY);
        GotoXY_(0, 22);
        printf("************* YOU HAVE WON!! *************");
        Pause();
        GotoXY_(0, 22);
        printf(szClear);
        GotoXY_(0, 22);
      } else {
        Board[UserX][UserY] = bVISITED;
        DisplayCell(UserX, UserY);
        UserX = NewX;
        UserY = NewY;
        Board[UserX][UserY] = bCURRENT;
        DisplayCell(UserX, UserY);
      }
    }
  }
  GotoXY_(0, GRID_Y*2+2);
  printf("Number of mines around you: %d", CountMines(UserX, UserY));
  GotoXY_(0, SCREEN_Y - 1);
  return (bAbort);
}
void PlaceUserMark(void)
{
  BOOL bDone, bAbort;
  int  Ch;
  int  NewX, NewY;
  int  dx, dy;

  bAbort = FALSE;
  GotoXY_(0, 24);
  printf("Mark in which direction? (ESC=abort)");
  bDone = FALSE;
  while (!bDone) {
    bDone = TRUE;
    Ch = getch();
    switch (Ch) {
      case 0:
        Ch = getch();
        switch (Ch) {
          case 71:  /* home */
            dx = -1;
            dy = -1;
            break;
          case 72:  /* up arrow */
            dx = 0;
            dy = -1;
            break;
          case 73:  /* page up */
            dx = 1;
            dy = -1;
            break;
          case 75:  /* left arrow */
            dx = -1;
            dy = 0;
            break;
          case 77:  /* right arrow */
            dx = 1;
            dy = 0;
            break;
          case 79:  /* end */
            dx = -1;
            dy = 1;
            break;
          case 80:  /* down arrow */
            dx = 0;
            dy = 1;
            break;
          case 81:  /* page down */
            dx = 1;
            dy = 1;
            break;
          default:
            bDone = FALSE;
            break;
        }
        break;
      case '7':  /* home */
        dx = -1;
        dy = -1;
        break;
      case '8':  /* up arrow */
        dx = 0;
        dy = -1;
        break;
      case '9':  /* page up */
        dx = 1;
        dy = -1;
        break;
      case '4':  /* left arrow */
        dx = -1;
        dy = 0;
        break;
      case '6':  /* right arrow */
        dx = 1;
        dy = 0;
        break;
      case '1':  /* end */
        dx = -1;
        dy = 1;
        break;
      case '2':  /* down arrow */
        dx = 0;
        dy = 1;
        break;
      case '3':  /* page down */
        dx = 1;
        dy = 1;
        break;
      case 27:
      case 13:
      case 10:
      case  8:
        bAbort = TRUE;
        break;
      default:
        bDone = FALSE;
        break;
    }
  }
  GotoXY_(0, 24);
  printf(szClear);
  if (!bAbort) {
    NewX = UserX + dx;
    NewY = UserY + dy;
    if ((NewX < 0) || (NewX >= GRID_X) || (NewY < 0) || (NewY >= GRID_Y)) {
      GotoXY_(0, 24);
      printf("ERROR: Out of bounds!!");
      Pause();
      GotoXY_(0, 24);
      printf(szClear);
    } else {
      GotoXY_(0, 24);
      if (UserMark[NewX][NewY] != 0) {
        Ch = 0;
      } else {
        Ch = 'M';
      }
      UserMark[NewX][NewY] = Ch;
      DisplayCell(NewX, NewY);
    }
  }
  GotoXY_(0, 24);
}
void ComputeAdjacency(int x, int y)
{
  int dX, dY;
  int BombCount;
  int Cell;
  if ((x >= 0) && (x < GRID_X) && (y >= 0) && (y < GRID_Y)) {
    if ((Board[x][y] == bVISITED) || (Board[x][y] == bCURRENT)) {
      BombCount = CountMines(x, y);
      Cell = 0;
      for (dX=-1; dX<=1; dX++) {
        for (dY=-1; dY<=1; dY++) {
          if (!((dX == 0) && (dY == 0))) {
            if ((x+dX >= 0) && (x+dX < GRID_X) &&
                (y+dY >= 0) && (y+dY < GRID_Y)) {
              if ((Board[x+dX][y+dY] != bVISITED) &&
                  (Board[x+dX][y+dY] != bCURRENT)) {
                if (UserMark[x+dX][y+dY] != 0) {
                  BombCount--;
                } else {
                  AdjacencyGroup[x][y].Cell[Cell][0] = x+dX;
                  AdjacencyGroup[x][y].Cell[Cell][1] = y+dY;
                  Cell++;
                }
              }
            }
          }
        }
      }
      AdjacencyGroup[x][y].BombCount = BombCount;
      AdjacencyGroup[x][y].CellCount = Cell;
    } else {
      AdjacencyGroup[x][y].CellCount = 0;
      AdjacencyGroup[x][y].BombCount = -1;   /** Don't look flag */
    }
  }
}
int AddToPositionList(int PositionList[GRID_X * GRID_Y][2],
                      int PositionListHead, int x, int y)
{
  int  nIndex;
  BOOL bFound;
  ComputeAdjacency(x, y);
  bFound = FALSE;
  for (nIndex=0; (nIndex<PositionListHead) && (!bFound); nIndex++) {
    if ((PositionList[nIndex][0] == x) && (PositionList[nIndex][1] == y)) {
      bFound = TRUE;
    }
  }
  if (!bFound) {
    PositionList[PositionListHead][0] = x;
    PositionList[PositionListHead][1] = y;
    PositionListHead++;
  }
  if (PositionListHead > GRID_X * GRID_Y) {
    GotoXY_(0, 22);
    printf("ERROR! PositionListHead > max (%d)", PositionListHead);
    Pause();
    GotoXY_(0, 22);
    printf(szClear);
     GotoXY_(0, 22);
   }
   return (PositionListHead);
}
int AddSurroundingToPositionList(int PositionList[GRID_X * GRID_Y][2],
                                 int PositionListHead, int x, int y)
{
  int dX, dY;

  for (dX=-1; dX<=1; dX++) {
    for (dY=-1; dY<=1; dY++) {
      if ((x+dX >= 0) && (x+dX < GRID_X) && (y+dY >= 0) && (y+dY < GRID_Y)) {
        if ((Board[x+dX][y+dY] == bVISITED) ||
            (Board[x+dX][y+dY] == bCURRENT)) {
          PositionListHead = AddToPositionList(PositionList, PositionListHead,
x+dX, y+dY);
        }
      }
    }
  }
  return (PositionListHead);
}
BOOL FindPositionInAG(ADJACENCYGROUP *pAG, int x, int y)
{
  int  nIndex;
  BOOL bFound;
  bFound = FALSE;
  for (nIndex=0; nIndex<pAG->CellCount; nIndex++) {
    if ((pAG->Cell[nIndex][0] == x) && (pAG->Cell[nIndex][1] == y)) {
      bFound = TRUE;
    }
  }
  return (bFound);
}
void MarkBombCell(int x, int y)
{
  UserMark[x][y] = 'M';
  DisplayCell(x, y);
  if (Board[x][y] != bBOMB) {
    GotoXY_(0, 22);
    printf("LOGIC ERROR: I tagged a phantom bomb @ (%d,%d).", x, y);
    Pause();
```

(continued on page 112)

The opportunity of a lifetime might be looking for you.

That's right—the career opportunity you've always dreamed about might be out there looking for you at this very moment. Or maybe a month from now—or a year from now. But how will it ever find you?

The Professional Registry of The Marlton Career Connection, Ltd. is electronically scanned by large and small corporations, government agencies and other institutions requiring professionals at all levels—from senior management positions to programmers—for permanent jobs to consulting assignments. And YOUR IDENTITY IS KNOWN ONLY TO US. Subscribers request contact with you through The Professional Registry, which provides contact information ONLY WITH YOUR PERMISSION.

There are two ways to register. You can use your computer to dial up ours and be guided in entering your credentials. Dial (301) 816-9210*. Alternatively, ask for a registration form to be mailed to you by calling (301) 816-9651. PLEASE, No Resumes.

Registration is FREE—SO ACT NOW! You have everything to gain.

Marlton Career Connection

*8 bits, no parity, one stop bit, 300/1200/2400 baud

CIRCLE NO. 412 ON READER SERVICE CARD

Uncompromising

No Surveys.
No Interviews.
No Jokes.
(well, sometimes...)

The C Users Journal™

Simply the best tool for C programmers. Period.

YES – I want the best C information Now!

☐ 1 year ($28) – Save 49% off our newsstand rate!
☐ 2 years ($52)
☐ 3 years ($75)
☐ Bill me
☐ Payment Enclosed

Mail to: The C Users Journal™
2601 Iowa
Lawrence, KS 66047

Name ____________________
Company ____________________
Address ____________________
City/State/Zip ____________________

Foreign subscribers: 1 yr — $46 US; 2 yr. — $86 US; 3 yr. — $124 US

For Even Faster Service:
CALL (913) 841-1631
(VISA/MC)

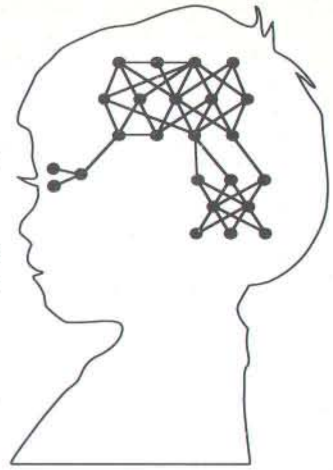
CIRCLE NO. 99 ON READER SERVICE CARD

Listing One *(Listing continued, text begins on page 72.)*

```
    GotoXY_(0, 22);
    printf(szClear);
    GotoXY_(0, 24);
  }
}
void VisitCell(int x, int y)
{
  if (Board[x][y] != bCURRENT) {
    if (Board[x][y] == bBOMB) {
      GotoXY_(0, 22);
      printf("LOGIC ERROR: I walked on a bomb @ (%d,%d).", x, y);
      Pause();
      GotoXY_(0, 22);
      printf(szClear);
      GotoXY_(0, 24);
    }
    Board[x][y] = bVISITED;
    DisplayCell(x, y);
  }
}
int CountCommonCells(ADJACENCYGROUP *pGroup1, ADJACENCYGROUP *pGroup2)
{
  int Cell, nCount;
  nCount = 0;
  for (Cell=0; Cell<pGroup1->CellCount; Cell++) {
    if (FindPositionInAG(pGroup2,
                         pGroup1->Cell[Cell][0], pGroup1->Cell[Cell][1])) {
      nCount++;
    }
  }
  return (nCount);
}
BOOL ProcessRule3(ADJACENCYGROUP *pCurrentAG, ADJACENCYGROUP *pTempAG,
                  int PositionList[GRID_X * GRID_Y][2],
                  int *pPositionListHead)
{
  int x;
  int BombCount, CellCount;
  int PositionListHead;
  int CellHolder[9][2];
  int CellHolderHead;
  BOOL bRetVal;
  PositionListHead = *pPositionListHead;
  bRetVal = FALSE;
  BombCount = pCurrentAG->BombCount;
  CellCount = pCurrentAG->CellCount;
  if (pTempAG->CellCount == CountCommonCells(pTempAG, pCurrentAG)) {
    BombCount -= pTempAG->BombCount;
    CellCount -= pTempAG->CellCount;
    if ((CellCount > 0) && ((BombCount == CellCount) || (BombCount == 0))) {
      bRetVal = TRUE;
      CellHolderHead = 0;
      CellCount = pCurrentAG->CellCount;
      for (x=0; x<CellCount; x++) {
        if (!FindPositionInAG(pTempAG, pCurrentAG->Cell[x][0],
                              pCurrentAG->Cell[x][1])) {
          if (BombCount == 0) {
            VisitCell(pCurrentAG->Cell[x][0], pCurrentAG->Cell[x][1]);
          } else {
            MarkBombCell(pCurrentAG->Cell[x][0], pCurrentAG->Cell[x][1]);
          }
          /* Queue up cells to put in position list for later */
          CellHolder[CellHolderHead][0] = pCurrentAG->Cell[x][0];
          CellHolder[CellHolderHead][1] = pCurrentAG->Cell[x][1];
          CellHolderHead++;
        }
      }
      for (x=0; x<CellHolderHead; x++) {
        PositionListHead = AddSurroundingToPositionList(
                           PositionList,
                           PositionListHead,
                           CellHolder[x][0],
                           CellHolder[x][1]);
      }
    }
  }
  *pPositionListHead = PositionListHead;
  return (bRetVal);
}
void EvaluatePosition(void)
{
  int  CurrentX, CurrentY;
  int  x, y;
  int  Cell;
  int  dX, dY;
  int  BombCount, CellCount;
  int  PositionList[GRID_X * GRID_Y][2], PositionListHead;
  ADJACENCYGROUP *pTempAG;
  BOOL bDone;
  BOOL bModifiedAny;
  bModifiedAny = TRUE;
  for (x=0; x<GRID_X; x++) {
    for (y=0; y<GRID_Y; y++) {
      ComputeAdjacency(x, y);
    }
  }
  PositionList[0][0] = UserX;
  PositionList[0][1] = UserY;
  PositionListHead = 1;
  while (bModifiedAny) {
    bModifiedAny = FALSE;
    while (PositionListHead > 0) {
      CurrentX = PositionList[0][0];
      CurrentY = PositionList[0][1];
      for (x=0; x<PositionListHead-1; x++) {
        PositionList[x][0] = PositionList[x+1][0];
        PositionList[x][1] = PositionList[x+1][1];
      }
      PositionListHead--;
      ComputeAdjacency(CurrentX, CurrentY);
      BombCount = AdjacencyGroup[CurrentX][CurrentY].BombCount;
      CellCount = AdjacencyGroup[CurrentX][CurrentY].CellCount;
      if ((CellCount > 0) && (BombCount > -1)) {
/*
           Rule 1: if number of bombs = number of cells, all are bombs!
*/
        if (CellCount == BombCount) {
          for (Cell=0; Cell<CellCount; Cell++) {
            x = AdjacencyGroup[CurrentX][CurrentY].Cell[Cell][0];
            y = AdjacencyGroup[CurrentX][CurrentY].Cell[Cell][1];
            MarkBombCell(x, y);
            PositionListHead = AddSurroundingToPositionList(PositionList,
                                                            PositionListHead,
                                                            x, y);

            bModifiedAny = TRUE;
          }
        } else {
/*
             Rule 2: if number of bombs = 0, all cells are ok!
*/
          if ((BombCount == 0) && (CellCount > 0)) {
            for (Cell=0; Cell<CellCount; Cell++) {
              x = AdjacencyGroup[CurrentX][CurrentY].Cell[Cell][0];
              y = AdjacencyGroup[CurrentX][CurrentY].Cell[Cell][1];
              VisitCell(x, y);
              PositionListHead = AddToPositionList(PositionList,
                                                   PositionListHead,
                                                   x, y);
              PositionListHead = AddSurroundingToPositionList(PositionList,
                                                              PositionListHead,
                                                              x, y);

              bModifiedAny = TRUE;
            }
          } else {
/*
               Rule 3: if AG completely overlaps another AG, subtract 2nd
                       # of bombs from 1st; check rules 1 & 2. If rule 1 or
                       2 is true in this case, stop looking in rule 3.
*/
            bDone = FALSE;
            for (Cell=0; (Cell<CellCount) && (!bDone); Cell++) {
              x = AdjacencyGroup[CurrentX][CurrentY].Cell[Cell][0];
              y = AdjacencyGroup[CurrentX][CurrentY].Cell[Cell][1];
              for (dX=-1; (dX<=1) && (!bDone); dX++) {
                for (dY=-1; (dY<=1) && (!bDone); dY++) {
                  if ((x+dX >= 0) && (x+dX < GRID_X) &&
                      (y+dY >= 0) && (y+dY < GRID_Y)) {
                    pTempAG = &AdjacencyGroup[x+dX][y+dY];
                    if (pTempAG->BombCount > 0) {       /* if == 0, no help! */
                      bDone = ProcessRule3(&AdjacencyGroup[CurrentX][CurrentY],
                                           pTempAG,
                                           PositionList,
                                           &PositionListHead);

                      if (bDone) {
                        bModifiedAny = TRUE;
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
    if (bModifiedAny) {
      for (x=0; x<GRID_X; x++) {
        for (y=0; y<GRID_Y; y++) {
          if ((Board[x][y] == bVISITED) || (Board[x][y] == bCURRENT)) {
            PositionListHead = AddToPositionList(PositionList,
                                                 PositionListHead,
                                                 x, y);

          }
        }
      }
    }
  }
}
BOOL LetUserMove(void)
{
  BOOL bDone;
  BOOL bQuit;
  int  Ch;
  bDone = FALSE;
  while (!bDone) {
    Ch = getch();
    switch (Ch) {
      case 0:
        Ch = getch();
        switch (Ch) {
          case 71:  /* home */
            bDone = Travel(-1, -1);
            break;
          case 72:  /* up arrow */
            bDone = Travel(0, -1);
            break;
          case 73:  /* page up */
            bDone = Travel(1, -1);
            break;
          case 75:  /* left arrow */
            bDone = Travel(-1, 0);
            break;
          case 77:  /* right arrow */
            bDone = Travel(1, 0);
            break;
          case 79:  /* end */
            bDone = Travel(-1, 1);
            break;
          case 80:  /* down arrow */
          bDone = Travel(0, 1);
          break;
        case 81:  /* page down */
          bDone = Travel(1, 1);
          break;
      }
      break;
    case '7':  /* home */
      bDone = Travel(-1, -1);
      break;
    case '8':  /* up arrow */
      bDone = Travel(0, -1);
      break;
    case '9':  /* page up */
      bDone = Travel(1, -1);
      break;
    case '4':  /* left arrow */
      bDone = Travel(-1, 0);
      break;
    case '6':  /* right arrow */
      bDone = Travel(1, 0);
      break;
    case '1':  /* end */
      bDone = Travel(-1, 1);
      break;
    case '2':  /* down arrow */
      bDone = Travel(0, 1);
      break;
    case '3':  /* page down */
      bDone = Travel(1, 1);
      break;
    case 'Q':
    case 'q':
    case 27:
      bDone = TRUE;
      break;
    case 'M':
    case 'm':
      PlaceUserMark();
      break;
    case '?':
      EvaluatePosition();
      break;
   }
  }
  bShowBombs = TRUE;
  PaintBoard();
  GotoXY (0, SCREEN_Y - 1);
  printf("Again (Y/n)? ");
  bDone = FALSE;
  while (!bDone) {
    Ch = getch();
    if ((Ch == 'Y') || (Ch == 'y') || (Ch == 13) || (Ch == 10)) {
      bDone = TRUE;
      bQuit = FALSE;
      printf("Y\n");
    }
    if ((Ch == 'N') || (Ch == 'n')) {
      bDone = TRUE;
      bQuit = TRUE;
      printf("N\n");
    }
    if (Ch == NULL) {
      getch();
    }
  }
  return (bQuit);
}
int main(int argc, char *argv[])
{
  BOOL bDone;
  Initialize();
  if ((argc > 1) &&
      (argv[1][0] == '/') &&
      ((argv[1][1] == 's') || (argv[1][1] == 'S'))) {
    bSafeGame = TRUE;
    printf("SAFE GAME in effect.\n");
  } else {
    bSafeGame = FALSE;
  }
  bDone = FALSE;
  while (!bDone) {
    SetUpBoard();
    bDone = LetUserMove();
  }
  return (0);
}
```

End Listing

"BrainMaker is the most fascinating computer software I've ever seen. This is hotter than hot."
John Dvorak
PC Magazine



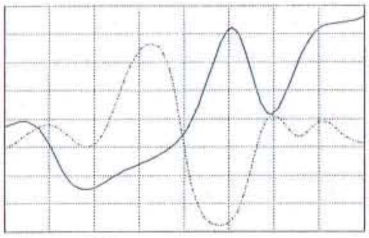
New! BrainMaker V2.0.

The #1 selling Neural Network.
Over 7,000 copies sold. Here's why:

Best documentation: 255 pg *Introduction to Neural Networks*, 332 pg Users Guide, 80 pg Application Guide. Our documentation includes **everything** you need to know to build networks and applications now.

Easiest to use: Pull down menus and dialog boxes. Supports mouse, color. NetMaker automatic network generation program helps you build networks in seconds.

Fastest:
128,000 cps on 286/8;
250,000 cps on 286/20;
500,000 cps on 386/33;
3 Million cps on C25;
4 Billion cps on N64.



Most complete product line:

- BrainMaker is $195, including all three books and NetMaker, our automated network generation program.
- BrainMaker graphics post-processor is $79.95.
- Hypersonic Training is $149.95. Our proprietary algorithm trains networks in minutes instead of hours.
- BrainMaker Professional is $795. Reads Dbase, Lotus, Excel, and Quattro files; includes graphics, binary files, hypersonic training, very large networks, and much more.
- BrainMaker C25 Accelerator from $2500. 3 million cps.
- Support for the Intel N64 neural network chip. You can build and train networks for this important new chip **now!**

Call or write for a Free Catalog!
California Scientific Software
160 E Montecito #E, Sierra Madre, CA 91024
(818) 355-1094

"I recommend BrainMaker without reservation." Steve Gibson, Infoworld

CIRCLE NO. 424 ON READER SERVICE CARD

Listing One *(Text begins on page 77.)*

```
/* eesc.c -- edge enhancement system in C */
#include <stdio.h>
#define ASize(x)         (sizeof(x)/sizeof(x[0]))         /* length of array */

/* PrintGraph() - print out graph of an array of numbers*/
FILE    *PFOutFp  = {stdout};
int PrintGraph( PFarray, ILen, Iny )
float            *PFarray;        /* pointer to floating piont array */
int               ILen;           /* length of the array */
int               Iny;            /* # of points along the y-axis */
{
    float         FMin, FMax;     /* minimum and maximum values */
    float         FSc, FOff;      /* scale & offset */
    int           Iwx;            /* work index */
    int           Ilx;            /* line index */
    int           ITx;            /* temp index */
    int           IpTx;           /* prior line index */
    int           Ich;            /* character to display */
    /* --- check that all parameters are "reasonable" --- */
    if ( PFarray == (float *)0 ¦¦ ILen <= 0 ¦¦ Iny <= 1 )
        return( -1 );
    /* --- compute minimum and maximum values for array --- */
    FMin = PFarray[0];
    FMax = PFarray[0];
    for( Iwx = 1; Iwx < ILen; Iwx++ ) {
        if ( FMin > PFarray[Iwx] )       FMin = PFarray[Iwx];
        if ( PFarray[Iwx] > FMax )       FMax = PFarray[Iwx];
    }
    if ( FMin > 0.0 ) FMin = 0.0;
    /* --- from minimum and maximum, compute scale and offset --- */
    if ( (FMax - FMin) < .0001 ) {
        /* --- assume that all values are the same --- */
        FSc  = 1.0;
        FOff = -FMin;
    } else {
        FSc  = Iny / (FMax - FMin);
        FOff = -FSc * FMin;
    }
    IpTx = 0;
    fputc( '\n', PFOutFp );
    for( Ilx = Iny; Ilx >= 0; Ilx-- ) {
        for( Iwx = 0; Iwx < ILen; Iwx++ ) {
            ITx = FSc * PFarray[Iwx] + FOff;
            if ( ITx < 0   ) ITx = 0;
            if ( ITx > Iny ) ITx = Iny;
            if ( Iwx == 0 ) IpTx = ITx;
            if ( (IpTx < Ilx && Ilx < ITx)  ¦¦
                 (ITx  < Ilx && Ilx < IpTx) ¦¦
                 (ITx == Ilx) )             Ich = 'x';
            else                            Ich = ' ';
            fputc( Ich, PFOutFp );
            IpTx = ITx;
        }
        fputc( '\n', PFOutFp );
    }
    return( 0 );
}
/* Convolve() - Convolve a filter with a one-dimensional signal */
int Convolve( PFilter, IFLen, PFInVec, PFResVec, ILen )
float            *PFilter;        /* pointer to filter coefficients */
int               IFLen;          /* number of coefficients in filter */
float            *PFInVec;        /* input signal vector */
float            *PFResVec;       /* output result vector */
int               ILen;           /* length of input & result vectors */
{
    int           IFx;            /* filter index */
    int           IResX;          /* result index */
    int           IResXLast;      /* index of last result item */
    int           IResXFirst;     /* index of first result item */
    double        DRv;            /* result value */
    /* --- check for things which do not make sense --- */
    if ( IFLen <= 0 ¦¦ ILen <= IFLen )  return( -1 );
    if ( PFilter == (float *)0 ¦¦
         PFInVec == (float *)0 ¦¦ PFResVec == (float *)0 ) return( -1 );
    /* --- convolve the filter with the signal --- */
    IResXFirst = IFLen / 2;
    IResXLast  = ILen - (IFLen-1)/2;
    for( IResX = IResXFirst; IResX < IResXLast; IResX++ ) {
        DRv =  0.0;
        for( IFx = 0; IFx < IFLen; IFx++ )
            DRv += PFilter[IFx] * PFInVec[IResX-IResXFirst+IFx];
        PFResVec[IResX] = DRv;
    }
    /* --- handle left edge specially --- */
    DRv = PFResVec[IResXFirst];
    for( IResX = 0; IResX < IResXFirst; IResX++ ) PFResVec[IResX] = DRv;
    /* --- likewise right edge --- */
    DRv = PFResVec[IResXLast-1];
    for( IResX = IResXLast; IResX < ILen; IResX++ )  PFResVec[IResX] = DRv;
    /* --- we are done --- */
    return( 0 );
}

/* NNCycle() - perform one iteration with Neural Network */
int NNCycle( Bias, PFilter, IFLen, PFInVec, PFResVec, ILen )
float             Bias;           /* bias for PE */
float            *PFilter;        /* pointer to filter coefficients */
int               IFLen;          /* number of coefficients in filter */
float            *PFInVec;        /* input signal vector */
float            *PFResVec;       /* output result vector */
int               ILen;           /* length of input & result vectors */
{
    int           IFx;            /* filter index */
    int           IResX;          /* result index */
    int           IResXLast;      /* index of last result item */
    int           IResXFirst;     /* index of first result item */
    double        DRv;            /* result value */
    /* --- check for things which do not make sense --- */
    if ( IFLen <= 0 ¦¦ ILen <= IFLen )  return( -1 );
    if ( PFilter == (float *)0 ¦¦
         PFInVec == (float *)0 ¦¦ PFResVec == (float *)0 ) return( -1 );
    /* --- convolve the filter with the signal --- */
    IResXFirst = IFLen / 2;
    IResXLast  = ILen - (IFLen-1)/2;
    for( IResX = IResXFirst; IResX < IResXLast; IResX++ ) {
        DRv =  -Bias;                                       /* NN special */
        for( IFx = 0; IFx < IFLen; IFx++ )
            DRv += PFilter[IFx] * PFInVec[IResX-IResXFirst+IFx];
        /* --- apply clamped linear transfer function to output --- */
        if (      DRv < 0.0 ) DRv = 0.0;                    /* NN special */
        else if ( DRv > 1.0 ) DRv = 1.0;                    /* NN special */
        PFResVec[IResX] = DRv;
    }
    /* --- handle left edge specially --- */
    DRv = PFResVec[IResXFirst];
    for( IResX = 0; IResX < IResXFirst; IResX++ ) PFResVec[IResX] = DRv;
    /* --- likewise right edge --- */
    DRv = PFResVec[IResXLast-1];
    for( IResX = IResXLast; IResX < ILen; IResX++ )  PFResVec[IResX] = DRv;
    /* --- we are done --- */
    return( 0 );
}

/* main() - main driver routine */
/* --- Input Signal --- */
float FSignal[] = {
  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,
  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,
  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.15,  0.20,  0.25,
  0.30,  0.35,  0.40,  0.45,  0.50,  0.55,  0.60,  0.65,  0.70,  0.75,
  0.80,  0.80,  0.80,  0.80,  0.80,  0.80,  0.83,  0.80,  0.70,  0.90,
  0.80,  0.80,  0.60,  0.90,  0.40,  0.60,  0.30,  0.10,  0.20,  0.20,
  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,
  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,
  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,
  0.20,  0.20,  0.20,  0.10,  0.25,  0.30,  0.10,  0.20,  0.20,  0.20,
  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20,  0.20 };

/* --- Result Signal --- */
float FResult1[ ASize(FSignal) ] = {0};
float FResult2[ ASize(FSignal) ] = {0};
/* --- Convolver for Neural Network --- */
float FMHF[] = {
 -0.10, -0.60, -0.30,  0.50,  1.10,  0.50, -0.30, -0.60, -0.10 };
/* --- Standard (sobel) edge detector --- */
float FSobel[] = { -1.0,  0.0,  1.0 };

main()
{
    int           Iwx;
    float        *PFResA, *PFResB, *PFSwap;
    PrintGraph( &FSignal[0], ASize(FSignal), 40 );
    fputs( "\n--- Original Signal ---\n\n", PFOutFp );
    Convolve( &FSobel[0], ASize(FSobel),
              &FSignal[0], &FResult1[0], ASize(FSignal) );
    PrintGraph( &FResult1[0], ASize(FResult1), 40 );
    fputs( "\n--- Result of applying sobel edge detector to image---\n\n",
        PFOutFp );
    PrintGraph( &FSignal[0], ASize(FSignal), 40 );
    fputs( "\n--- Original Signal ---\n\n", PFOutFp );
    PFResA = &FSignal[0];
    PFResB = &FResult1[0];
    PFSwap = &FResult2[0];
    for( Iwx = 1; Iwx <= 8; Iwx++ ) {
        NNCycle( .02, &FMHF[0], ASize(FMHF), PFResA, PFResB, ASize(FSignal) );
        PrintGraph( PFResB, ASize(FResult1), 40 );
        fprintf( PFOutFp, "\n--- Cycle number %d ---\n\n", Iwx );
        PFResA = PFResB;                     /* swap result pointers */
        PFResB = PFSwap;
        PFSwap = PFResA;                     /* next ResB */
    }
    exit( 0 );
}
```

End Listing One

Listing Two

```
Neural Network Based "Edge Enhancement System"                                    1
  Written by:  Casimir C. "Casey" Klimasauskas                                    2
  January 6, 1990                                                                 3
  Lotus 1-2-3 version 3.0 spreadsheet                                             4
                                                                                  5
                                                                                  6
                    0.020 Bias                                                    7
                                                                                  8
      Low    Low             Raw                                                  9
     Pass   Pass     MHF   Input                                                 10
    Output Filter Filter  Data                                                   11
                                                                                 12
Iteration                          0     1    2    3    4    5    6    7    8    13
Graph     A                        B     C         D         E              F    14
                                                                                 15
                                0.20  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    16
                                0.20  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    17
                                0.20  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    18
                                0.20  0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    19
```

(Listing continued on page 116)

SOFTWARE DEVELOPERS: ARE YOUR PRODUCTS TOO HARD TO INSTALL?

you are still using *Ye Olde INSTALL.BAT File* to install your product, there is a much etter way. Let's face it, the computer industry is maturing. What was entirely dequate a few years ago is unacceptable today. If you have been requiring your nd-users to use batch files, COPY, DISKCOPY, or RESTORE to install your distribution disks, ou are making an unfavorable first impression with them. Not only are these methods rimitive, they have virtually no resilience to errors, and they rarely look professional.

here is an alternative...

NSTALL 3.0 will not only elegantly conquer your installation problems, it can also greatly ncrease your end-user's confidence in your product by automatically checking the ntegrity of the distribution disk(s).

NSTALL includes a 100% **reliable** CONFIG.SYS and AUTOEXEC.BAT file editor which can sk for permission before creating/editing existing files, and always creates backup files.

- Multiple source and target disks, multiple target subdirectories (which *INSTALL* will create if necessary).
- Intelligently handles all DOS errors including open disk drive doors, unformatted and full disks, write protected disks, network violation errors, bad disk sectors, insufficient memory, etc.
- Forget long sections of text in your documentation covering installation procedures. Instead, the installation section can now simply say "TYPE A:INSTALL" and *INSTALL* handles the rest.
- You can allow your end-users to only install parts of your product such as various device drivers.
- Fast, full-screen, text oriented windows interface.
- Quick, clean method of adding-in new features.
- High performance data compression will typically cut your number of distribution disks in half.
- Utilizes all available RAM for lightning-fast file transfers.
- *INSTALL* can create a single self-extracting (.EXE) file containing all of your distributions files.
- Installation logic may be based on the monitor (B&W/MONO/CGA/EGA/VGA), CPU (8086/268/386), DOS version, ANSI.SYS, NetBIOS, LIM, 80x87, LPT/COM port presence, disk capacity, disk free space, etc. to allow you to install a highly machine-specific product without burdening the end-user with questions s/he may not understand.
- Files up to 4.3 Gigabytes in size may be distributed by splitting the file across multiple distribution disks.
- No royalties. Unlimited free technical support. No programming needed!

ncludes C source code, make, project, and link files. 30-day money-back guarantee.

NSTALL offers 4 years of proven reliability that has been used by thousands of developers in 23 countries to install millions of copies of software including some of the nost prestigious products in the industry. We would like to add your name to that list.

Call or FAX before <u>4:00 today</u> and receive your order <u>tomorrow</u>! $249.95 + Shipping

KNOWLEDGE DYNAMICS CORPORATION

Highway Contract 4 Box 185-H, Canyon Lake, Texas 78133
MasterCard/VISA/COD/POs Welcome

1-800-331-2783 (Sales)
1-512-964-3994 (International)
1-512-964-3958 (24-hr FAX)
1-512-964-3929 (24-hr BBS)

SLATE **Universal Printer Driver**

Would support for over 400 printers give you a competitive edge?

By including SLATE, you can provide **immediate support for over 400 printers.** SLATE comes with its own **screen based configuration system** so you don't need to worry about end user setup.

You can use SLATE in your product with **no royalties.**

Make your product more functional and competitive by taking advantage of SLATE's **advanced features:**

- Support multiple printers on the same system.
- Output to parallel printers, serial printers, DOS files, console, DOS print spooler, and Novell network printers.
- Include laser printer soft fonts.
- Support proportional fonts.
- Set exact print positions.
- Color printing.
- Kerning, leading, overstrike, underlining, and strike through.

You can use advanced features of modern printers while working smoothly with printers without those features.

What is SLATE?

SLATE is a set of C libraries with over 150 text printing functions, a Database of over 400 printers, and tools for end user configuration and testing.

Support over 400 printers for less cost than developing one more driver. **Call now to order SLATE for only $299** with our risk free, 30 day return policy.

We accept Visa, Mastercard, COD's or PO's from qualified companies. Source code, maintenance, and site licenses are available. Also ask for information about our S_PRINT Text Formatting System for Software Developers.

800-346-3938

The Symmetry Group
PO Box 26195
Columbus, OH 43226
614-431-2667
FAX 614-431-5734

CIRCLE NO. 237 ON READER SERVICE CARD

Listing Two *(Listing continued, text begins on page 77.)*

```
     0.00    0.00   -0.10    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     20
     0.00    0.00   -0.60    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     21
     0.00    0.00   -0.30    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     22
     0.00   -1.00    0.50    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     23
     0.00    0.00    1.10    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     24
     0.00    1.00    0.50    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     25
     0.00    0.00   -0.30    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     26
     0.00    0.00   -0.60    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     27
     0.00    0.00   -0.10    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     28
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     29
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     30
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     31
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     32
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     33
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     34
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     35
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     36
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     37
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     38
     0.00                    0.20   0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00     39
     0.00                    0.20   0.03 0.02 0.00 0.00 0.00 0.00 0.00 0.00     40
     0.00                    0.20   0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00     41
    -0.05                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     42
     0.00                    0.15   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     43
     0.10                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     44
     0.10                    0.25   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     45
     0.10                    0.30   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     46
     0.10                    0.35   0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00     47
     0.10                    0.40   0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00     48
     0.10                    0.45   0.02 0.00 0.00 0.00 0.00 0.00 0.00 0.00     49
     0.10                    0.50   0.03 0.00 0.00 0.00 0.00 0.00 0.00 0.00     50
     0.10                    0.55   0.03 0.00 0.00 0.00 0.00 0.00 0.00 0.00     51
     0.10                    0.60   0.04 0.00 0.00 0.00 0.00 0.00 0.00 0.00     52
     0.10                    0.65   0.05 0.00 0.00 0.00 0.00 0.00 0.00 0.00     53
     0.10                    0.70   0.09 0.00 0.00 0.00 0.00 0.00 0.00 0.00     54
     0.10                    0.75   0.15 0.12 0.18 0.28 0.46 0.76 1.00 1.00     55
     0.05                    0.80   0.17 0.20 0.31 0.49 0.79 1.00 1.00 1.00     56
     0.00                    0.80   0.15 0.12 0.16 0.26 0.43 0.71 1.00 1.00     57
     0.00                    0.80   0.10 0.00 0.00 0.00 0.00 0.00 0.00 0.00     58
     0.00                    0.80   0.05 0.00 0.00 0.00 0.00 0.00 0.00 0.00     59
     0.00                    0.80   0.06 0.00 0.00 0.00 0.00 0.00 0.00 0.00     60
     0.03                    0.80   0.13 0.06 0.03 0.00 0.00 0.00 0.00 0.00     61
     0.00                    0.83   0.06 0.00 0.00 0.00 0.00 0.00 0.00 0.00     62
    -0.13                    0.80   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     63
     0.10                    0.70   0.01 0.00 0.00 0.00 0.00 0.00 0.00 0.00     64
     0.10                    0.90   0.21 0.08 0.06 0.01 0.00 0.00 0.00 0.00     65
    -0.10                    0.80   0.18 0.14 0.00 0.00 0.00 0.00 0.00 0.00     66
    -0.20                    0.80   0.22 0.05 0.00 0.00 0.00 0.00 0.00 0.00     67
     0.10                    0.60   0.13 0.05 0.00 0.00 0.00 0.00 0.00 0.00     68
    -0.20                    0.90   0.29 0.27 0.32 0.50 0.78 1.00 1.00 1.00     69
    -0.30                    0.40   0.26 0.28 0.38 0.58 0.93 1.00 1.00 1.00     70
    -0.10                    0.60   0.11 0.04 0.05 0.13 0.26 0.50 0.73 0.99     71
    -0.50                    0.30   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     72
    -0.10                    0.10   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     73
     0.10                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     74
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     75
     0.00                    0.20   0.05 0.04 0.03 0.02 0.01 0.00 0.00 0.00     76
     0.00                    0.20   0.01 0.02 0.02 0.02 0.01 0.00 0.00 0.00     77
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     78
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     79
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     80
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     81
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     82
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     83
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     84
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     85
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     86
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     87
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     88
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     89
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     90
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     91
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     92
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     93
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     94
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     95
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     96
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     97
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     98
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00     99
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    100
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    101
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    102
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    103
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    104
     0.00                    0.20   0.01 0.02 0.02 0.01 0.00 0.00 0.00 0.00    105
     0.00                    0.20   0.06 0.04 0.02 0.00 0.00 0.00 0.00 0.00    106
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    107
    -0.10                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    108
     0.05                    0.10   0.00 0.00 0.00 0.00 0.02 0.05 0.08 0.12    109
     0.20                    0.25   0.08 0.13 0.19 0.28 0.43 0.67 1.00 1.00    110
    -0.15                    0.30   0.12 0.14 0.20 0.29 0.45 0.71 1.00 1.00    111
    -0.10                    0.10   0.00 0.00 0.00 0.02 0.06 0.13 0.25 0.41    112
     0.10                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    113
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    114
     0.00                    0.20   0.05 0.03 0.00 0.00 0.00 0.00 0.00 0.00    115
     0.00                    0.20   0.01 0.02 0.01 0.00 0.00 0.00 0.00 0.00    116
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    117
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    118
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    119
     0.00                    0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    120
                             0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    121
                             0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    122
                             0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    123
                             0.20   0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00    124
                                                                               Row
A       B       C       D       E    F    G    H    I    J    K    L    M   Column
```

End Listings

C CODE FOR THE PC

source code, of course

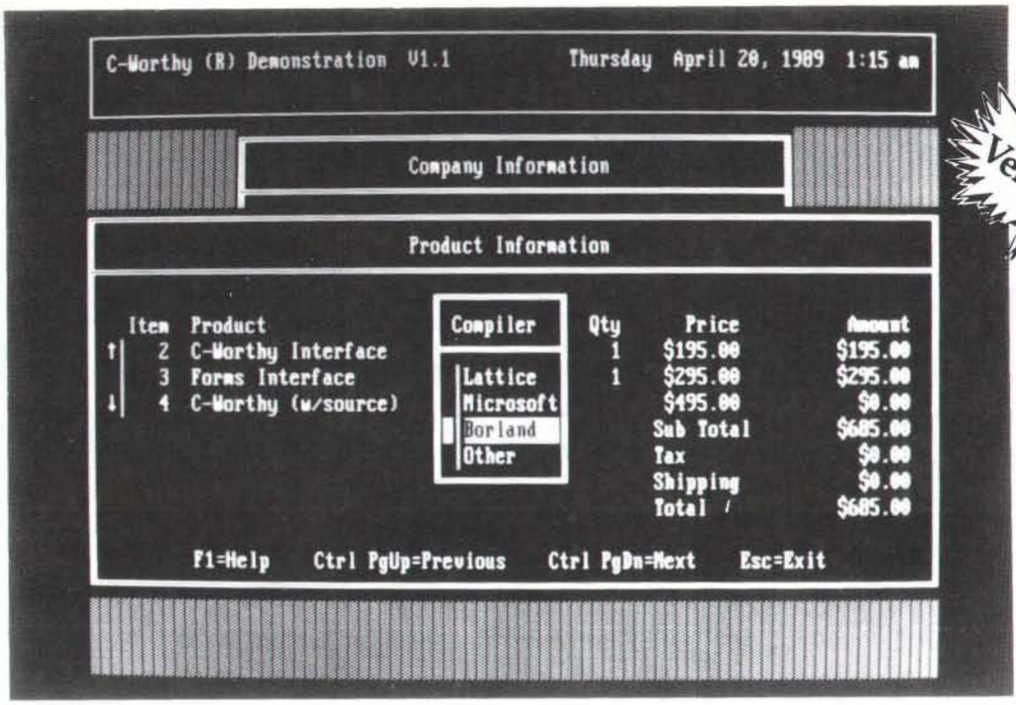
Updated!	MS-DOS File Compatibility Package (create, read & write MS-DOS file systems on non-MS-DOS computers)	$750
	CQL Query System (SQL retrievals on B-trees plus windows)	$325
	GraphiC 5.0 (high-resolution, DISSPLA-style scientific plots in color & hardcopy)	$325
	PC Curses (Aspen, Software, System V compatible, extensive documentation)	$290
	C-Data Manager (object-oriented data management, persistent objects from runtime definitions, network and entity models)	$250
	MEWEL (extensible window and event library by Magma Software; message-passing & object-oriented; SAA-compatible; dialog editor)	$250
	TurboTEX (Release 2.0; HP, PS, dot drivers; CM fonts; LaTEX; MetaFont)	$250
	db_File & db_Retrieve by Raima (B-tree and network database with SQL query and report writer; multi-user $475)	$245
	Greenleaf Communications Library (interrupt mode, modem control, XON-XOFF; specify compiler)	$225
	CDirect (multi-user hashed file manager; variable length fields, binary or ASCII data, alternate keys)	$210
	SilverComm (complete asynchronous communications library)	$210
NEW!	Wendin-DOS Plus (self-bootable, multitasking, multiuser MS-DOS replacement; includes XTC editor)	$180
	QuickGeometry Library (large collection of mathematics, graphics, display & DXF subroutines for CAD/CAM/CAE/CNC)	$170
	CBTree (B+tree ISAM driver, multiple variable-length keys)	$165
	TurboGeometry (library of routines for computational geometry, Version 3.0)	$160
	AT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for ATs)	$160
	WKS Library Version 2.01 (C program interface to Lotus 1-2-3, dBase, Supercalc 4, Quatro, & Clipper)	$155
	OS/88 (industrial-strength U**x-like operating system, many tools, cross-development from MS-DOS)	$150
	Cephes Mathematical Library (over 100 high-quality, double-precision scientific functions)	$150
	ME Version 2.1 (programmer's editor with C-like macro language by Magma Software; Version 1.31 still $75)	$140
	Vmem/C (virtual memory manager; least-recently used pager; dynamic expansion of swap file)	$140
	Turbo G Graphics Library (all popular adapters, hidden line removal)	$135
	Rogue Wave Vector & Matrix Classes (inc. C++ overloadings for standard operators, matrix inversion & FFT; Zortech or GNU C++)	$125
	Power Search by Blaise Computing (regular-expression compiler; generates machine code on the fly)	$120
	Install 2.3 (automatic installation program; user-selected partial installation; CRC checking)	$120
	TE Editor Developer's Kit (full screen editor, undo command, multiple windows)	$115
NEW!	Hold Everything (spawn new programs; swap parent to EMS or disk; handles video, interrupts, & environment; returns error level)	$105
	B-Strings (dynamic string handling; cut, copy, paste, search, user input, etc.; non-fragmenting memory management)	$105
	Minix Operating System (Version 1.3; U**x-like operating system, includes manual)	$105
	PC/IP (CMU/MIT TCP/IP for PCs; Ethernet, Appletalk & NETBIOS drivers, RVD, gateways)	$100
	B-Tree Library & ISAM Driver (file system utilities by Softfocus)	$100
	The Profiler (program execution profile tool)	$100
	QC88 C compiler (ASM output, small model, no longs, floats or bit fields, 80+ function library)	$90
Cheaper!	Booter Toolkit (floppy disk bootstrap routines, DOS file system, light-weight multitasking, windows, fast memory management)	$85
	Otter 1.0 (beautiful theorem-prover by Bill McCune; includes manual & two books by Wos; complete starter kit)	$80
	JATE Async Terminal Emulator (includes file transfer and menu subsystem)	$80
NEW!	PowerSTOR (extended heap space on extended memory, expanded memory, and/or hard disk)	$80
	MultiDOS Plus (DOS-based multitasking, intertask messaging, semaphores)	$80
NEW!	HY-PHEN-EX (a hyphenator for American English with over 4,800 rules)	$75
	Make (macros, all languages, built-in rules)	$75
	evalO (C function to evaluate ASCII infix expression string; 17 built-in functions)	$75
	XT BIOS Kit (roll your own BIOS with this complete set of basic input/output functions for XTs)	$75
	Professional C Windows (lean & mean window and keyboard handler)	$70
	Heap Expander (virtual memory manager using expanded memory, extended memory, and disk space)	$65
	Quincy (interactive C interpreter)	$60
	Symtab/Ptree (general-purpose symbol table/parse tree construction and management package; specify Symtab or Ptree)	$60
	Coder's Prolog (Version 3.0; inference engine for use with C programs)	$60
	Async-Termio (Unix V compatible serial interface for MS-DOS; stty, ioctl, SIGINT, etc.)	$55
	Backup & Restore Utility by Blake McBride (multiple volumes, file compression & encryption)	$50
	Floppy TAR (TAR backup and restore on MS-DOS devices; direct access to non-standard devices)	$50
	SuperGrep (exceptionally fast, revolutionary text searching algorithm; also searches sub-directories)	$50
	OBJASM (convert .obj files to .asm files; output is MASM compatible)	$50
	Multi-User BBS (chat, mail, menus, sysop displays; does not include Hayes modem driver)	$50
	LaplaceB (LaPlace polynomials, real and complex)	$50
	CLIPS (rule-based expert system generator, Version 4.3; advanced manuals available)	$50
NEW!	PCHRT (40 functions to manage multiple microsecond timers; generate precion delays; insert timers on any interrupt)	$45
	Kier DateLib (all kinds of date manipulation; translation, validation, formatting, & arithmetic)	$45
	Fortran-to-C Translator by Polyglot (Fortran-IV-like Fortran to ugly C; plan to adapt to your own flavor of Fortran)	$40
	DES Encryption & Decryption (2500 bits/second on 4.77 MHz PC for on-the-fly encryption at 2400 baud)	$40
	FlexList (doubly-linked lists of arbitrary data with multiple access methods)	$40
	Virtual Memory Manager by Blake McBride (LRU pager, dynamic swap file, image save/restore)	$40
	Heap I/O (treat all or part of a disk file as heap storage)	$40
	Bison & BYACC (YACC workalike parser generators; documentation; no restrictions on use of BYACC output)	$35
	PC-XINU (Comer's XINU operating system for PC)	$35
	RXC & EGREP (Regular Expression Compiler and Pattern Matching; RXC makes finite state machine from regular expression)	$35
Cheaper!	REGX Plus (search and replace string manipulation routines based on regular expressions)	$30
	CCALC (handy extended-precision calculator; real and complex models; many built-in functions)	$30
	GNU Awk & Diff for PC (both programs in one package)	$30
Updated!	6-Pack of Editors (baker's half-dozen public domain editors for use, study & hacking; includes microEmacs 3.10 & Stevie, a vi clone)	$30
	Crunch Pack (14 file compression & expansion programs)	$30
	Pascal P-Code Compiler & Interpreter or Pascal-to-C Translator (Wirth standard Pascal)	$25
NEW!	PC-MAIL (UUCP mailer by Wietse Z. Venema; send, receive, and manage UUCP mail)	$25
	FLEX (fast lexical analyzer generator; new, improved LEX; official BSD Version 2.1 with docs)	$25
	List-Pac (C functions for lists, stacks, and queues)	$25
	Using C++ Library (the code from the book by Bruce Eckel and then some; Zortech 2.0 compatible)	$25
	A68 (68000 cross-assembler)	$20
	XLT Macro Processor (general purpose text translator)	$20
	Data	
	Moby Pronunciator (150,000 words & phrases encoded with full IPA pronunciation & emphasis points; 900 distinguished by part-of-speech)	$160
	Moby Part-of-Speech (200,000 words and phrases described by prioritized part(s)-of-speech)	$120
	Moby Hyphenator (150,000 words fully hyphenated/syllabified)	$105
	Moby Words (500,000 words & phrases, 9,000 stars, 15,000 names)	$65
	Smithsonian Astronomical Observatory Subset (right ascension, declination, & magnitude of 258,997 stars)	$60
	U. S. Cities (names & longitude/latitude of 32,000 U.S. cities and 6,000 state boundary points)	$35
	The World Digitized (100,000 longitude/latitude of world country boundaries)	$30
	KST Fonts (13,200 characters in 139 mixed fonts: specify TEX or bitmap format)	$30
Updated!	Interactive Computer Ephemerii (high-precision moon, sun, planet & star positions; USNO (no source) & Downey 4.8 (C source))	$30
	U. S. Map (15,701 points of state boundaries)	$15

The Austin Code Works
11100 Leafwood Lane
Austin, Texas 78750-3409 USA

info@acw.com

Voice: (512) 258-0785
BBS: (512) 258-8831
FAX: (512) 258-1342

Free surface shipping for cash in advance **For delivery in Texas add 7%** **MasterCard/VISA**



```
C-Worthy (R) Demonstration  V1.1          Thursday  April 20, 1989  1:15 am

                        Company Information

                        Product Information

   Item  Product                  Compiler    Qty     Price         Amount
 ↑|   2  C-Worthy Interface                    1     $195.00       $195.00
  |   3  Forms Interface         |Lattice      1     $295.00       $295.00
 ↓|   4  C-Worthy (w/source)     |Microsoft          $495.00         $0.00
                                 |Borland            Sub Total     $685.00
                                 |Other              Tax             $0.00
                                                     Shipping        $0.00
                                                     Total /       $685.00

        F1=Help    Ctrl PgUp=Previous    Ctrl PgDn=Next    Esc=Exit
```

Focus your user on the next step by creating clean, attractive screens. Use C-Worthy's windows, hidden sub-menu, field validation, mouse support, field specific help, and much more.

Only C-Worthy® protects you from the trap that keeps 50% of the market from using your software . . .

The Trap: Users will reject your application if the interface is hard to use. They'll never even see the power of your program. "If you've messed up the interface, the product is probably dead in the water."

— Jeffrey Tarter, Editor, Soft•Letter

The Solution: Use C-Worthy's 30 highest-level functions to create applications for windows, menus, editing, help, error handling and forms. Or, use a larger mix of 430+ tested, documented, stable functions to finely tailor your work. Save one month or more in work effort while making a more thorough and consistent interface for a C development project. C-Worthy will produce an interface that pleases even the most demanding user.

Functionality Superior to Competing Environments

Screens and Windows Exploding, tiled or overlapping. Direct video access or virtual. Keep up to 50 active.

Menus Pop-up, Lotus style, pull-up and pull-down (windows style). Automatically sized to fit your choice of options, header, border type. Auto-select based on menu-item hot-key character you select.

Forms. Optional Form Interface Library speeds construction of data input forms, lets you apply latest research findings to gather and validate input. 23 field types, built-in or custom validation. Provide help at form and field levels. User-controllable movement sequence within the form.

Comprehensive Documentation 1400 Pages, fully indexed, plus 10,000+ lines of ready-to-go code that you can run, modify and learn from.

"C-Worthy is a higher level approach to human interface than other C libraries . . . Our software engineering department does not use another human interface library."

— **Kevin Kingdon, Senior Software Engineer, Novell, Inc.** (Over 500,000 users of Novell NetWare Utilities use a C-Worthy based interface every day.)

Version 1.1 is even easier to use

New 250 page manual: *The Programmer's Guide* will help you quickly choose the function that solves your problem.
Fully integrated mouse support: Your user can now use a mouse to select from menus or lists, move around a form, or edit text.
Plus: extended error handling, speaker control, scrolling forms, pop-up subforms, and new field types like: visible lists, unsigned long integer, matrix menu, more.

C-Worthy Interface Library

Object only	$195	Form Interface add-on	$100
Object with Forms	$295	Object with Forms & Source Code	$495
Call for OS/2 Prices		Call for International Pricing	

Ask for Your FREE Bonus Just for Trying C-Worthy with a 60 Day Moneyback Guarantee.

Call 800-821-2492 ext. 372
(in MA and outside U.S. 617-337-6963)

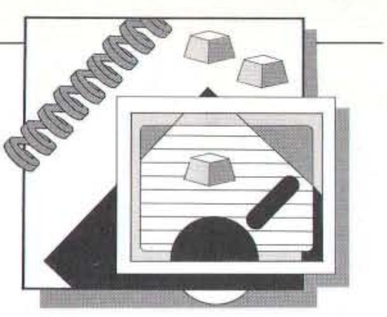
BONUS *Guidelines for Designing User Interface Software.* Contains 944 guidelines, including graphic functions and data display. 496 pages, $38.95.

Solution Systems™

541 Main Street, Suite 410
South Weymouth, MA 02190
617-337-6963
Fax: 617-337-7719

C-Worthy is a registered trademark of Custom Design Systems, Inc.

CIRCLE NO. 325 ON READER SERVICE CARD



Wrapping Up Software Development '90

"Can you imagine — the Berlin Wall breached! Free elections in Poland and Czechoslovakia! The Soviet leader proclaiming freedom of religion while visiting the pope! The Romanian dictator getting frosted on Christmas Day! Bush found the whole sequence of events unbelievable, dizzying, even a little bit frightening — and yet, unquestionably, the fat lady had opened her mouth and was hitting high C."

— Jamie Malinowski

"Gee," the president marveled in his trademark nasal whine and 1950s sixth-grade vocabulary, "it seems like it was only yesterday that their Premier was making trouble all the time and saying he was gonna bury us and stuff, and now their whole darn evil empire looks like it's coming apart. Just look at what's happened with their Mac products and Turbo Modula-2 and Turbo Basic. And what was that business at the SD '90 conference about Wallsoft's C compiler? Why'd they let a competitor speak at their press conference? Well, they must have got some bad Perrier down in Scotts Valley.

"It's pretty neat for us guys, that's for

Michael Swaine

sure," he told Steve "Wally" Ballmer, his squeaky-voiced confidante. "Now maybe the press will talk about that instead of always picking on us about our problems here at home or our problems with our so-called buddies at IBM." Blood brother oaths sworn in the childhood of the industry didn't seem to mean much in these grown-up days.

Chastened by this thought of the challenges that still lay before him, the president frowned at his long-time friend. "Gee. It's tough being big, Wally," he said.

Drugs, Bugs, and the DoD

Having joked about bugs as drugs in my February "Flames," I was interested to see that familiar *DDJ* author Do-While Jones was speaking about debuggers as a drug at Miller-Freeman's Software Development '90 in February.

"When you get sick," Jones said, "you should take medicine until you get well. Then you should stop." When your software is sick, he reasoned, you should use a debugger to find the problem. When you find the problem, you should put the debugger back on the shelf.

Debugger abuse, Jones says, comes from using a debugger when the program really isn't sick. Sick programs include those inherited in a messed-up state from someone else, and your own programs when they quit working suddenly, perhaps due to a change in the hardware or compiler or operating system. If something like this isn't wrong with the program, you shouldn't use a debugger, he says, because you will get hooked, starting a cycle of abuse: Dependence on a debugger makes for weak programmers who write sick software that can't be debugged without a debugger.

His point is that you usually don't need a debugger, that your knowledge, skills, and insight into the structure of the code constitute the best possible debugger. But you can't use your knowledge of the structure of code that has none. Writing well-structured programs is the prerequisite to rediscovering "the lost art of debugging."

As Jones describes it, the lost art is one of analyzing data flow. In a hierarchically structured program, modules at one level invoke modules at the next lower level, passing data to them, typically as parameters, getting data back, possibly as function values. Debugging a program consists of examining this data flow. The techniques he describes shouldn't be a revelation to any *DDJ* reader, but the kind of rigor he promotes in their use is perhaps uncommon.

You can test the data flow into and out of a module by writing a driver for the module. The driver sends selected data to the module and examines the results. This tests the module in isolation, but doesn't reflect its interaction with other modules.

For this, you need an integration test driver that calls the module one level higher than the modules you want to test. Data values are fed to it in order to check the interaction of the modules it invokes.

Finally, you can substitute a diagnostic module for any module in your program. A diagnostic module has the same name and parameters as the real module, but has a body that is simple and diagnostic. That is, it isn't so complicated that it could reasonably be a source of errors, and it does nothing but give useful information on the data-passing structure of which the module is a part.

In making his point, Jones used the word "hacker" to refer to one who pokes at the problem more or less at random until finding a fix that works. The word has a range of meanings, some good, some not, and this use is a legitimate use from the pejorative end of the word's meaning range. It was interesting, then, to find that he

Programmers Wholesaler™

Attention!
Corporate Accounts
Resellers
Programmers

Check our values!

	LIST	1-2	3+
BASIC			
Turbo Basic	100	67	64
QuickBASIC	99	67	64
Basic Dev. Sys. 7.0	495	329	321
C LANGUAGE - COMPILERS			
Lattice C - 6.0	250	156	143
Microsoft C 5.1	450	287	283
Microsoft Quick C	99	67	64
Turbo C by Borland	150	98	94
DATABASE MANAGEMENT			
Clarion	845	519	499
Paradox 3.0	725	489	479
DBASE			
Clipper Summer '87	695	429	419
dBASE IV	795	489	479
FoxBASE + 2.1	395	209	199
DBASE TOOLS			
Clear+ for dBASE	200	149	139
dBRIEF w/BRIEF	285	Save	Save
dSalvage	100	83	79
R&R Relational Reportwriter	149	99	93
EDITORS			
BRIEF	199	Save	Save
Epsilon	195	139	119
FORTRAN			
MS FORTRAN	450	299	289
OBJECT-ORIENTED			
Smalltalk/V	100	59	54
Zortech C++	199	Call	Call
OTHER PRODUCTS			
Carbon Copy Plus	195	115	104
HEADROOM by Helix	130	85	79
Norton Utilities Advanced	150	89	87
PC Tools Deluxe	129	85	79
Remote2	195	104	99
SPREADSHEETS			
1-2-3	495	299	289
Excel	495	339	329
TEXT SCREENS ADDONS			
C Worthy w/Forms	295	Save	Save
Greenleaf DataWindows	395	249	239
Vermont Views	495	319	299
WORD PROCESSING			
Sprint	200	134	129
WordPerfect	495	399	385

Prices subject to change without notice. "DD490W"

Programmers Wholesaler™

800-228-3736

CANADA
800-344-2495
FAX 617-740-1892

South Shore Park
P.O. Box 534
Accord, MA 02018
Hours: M-F 8:30-5

CIRCLE NO. 199 ON READER SERVICE CARD

espoused a distinctly hackerish philosophy in another talk at SD '90.

This was a discussion of the difference between engineers and computer scientists. Most of what he said was uncontroversial: Engineers are typically better educated in the physical sciences, computer scientists better grounded in a variety of programming languages. But the key differences, he said, are philosophical. Computer scientists and engineers have different ideas of economy; engineers are trained to look at the whole system, while computer scientists see the hardware as the platform, that is, as a given; computer scientists want to apply the best algorithm for the task (and are better equipped to do so), while engineers let the task dictate criteria for acceptability of an algorithm.

Jones is an engineer, and presented the engineering approach as the more hackerish, the more ad hoc of the two: Solve the problem no matter what. Here are some of the kinds of ad hoc, hackerish things Jones was saying:

- Most embedded systems don't need any kind of operating system.
- Data structures aren't important if you only have a few variables and RAM is limited.
- Tools are overrated.

Taken together, the talks make a point at a higher level. On the one hand, structure your code so that you can debug it by hand, and structure your approach to debugging. On the other, get the job done, meet the target launch date, satisfy the client. Together, the talks say simply, suit the tactic to the task. Every programmer should have a rich collection of ways of approaching a problem — a full set of intellectual tools. Structure is essential but so is flexibility.

Flexibility is a good trait for any software developer to cultivate. Do-While Jones is a Department of Defense engineer, like many who push bits for a living. One speaker at SD '90, William Roetzheim, speculated that half the software engineers in America were now or would be at some time in the future writing to DoD specs. Well. I don't relish the thought of anyone losing his or her job, but I am skeptical. This would not seem to be the best time to begin learning Ada.

The Big Challenge of Programming in the Large

"Stop talking of war cause we've heard it all before. Why don't you go out there and do something useful?"
— Sinead O'Connor

The image: The development team as bureaucracy, ideas trampled under political arguments, progress held up by mandatory progress reports, brilliant developers brought down to the level of their feeblest teammates.

The reality: The same thing, all too often. There appear to be excellent reasons to fear and to loathe the development team, and to long for the freedom to just write the damn thing. We all know the stories of the lone programmers who created masterpieces. It may be unrealistic to think that large software projects can be done any other way than through development teams, but it can be an inviting fantasy.

Knowledge, skills, and insight into the structure of the code constitute the best possible debugger

And yet, there are the stories of the team that worked. And most have even experienced that golden time, when the team fed off each others' talents and the result was a collaboration none of the participants could have done alone, and that all were proud of. OK, maybe it was only the Pringles can sculpture that we built on the Dean's front porch when we were freshmen, but when it comes to that sense of shared ownership of the work, is building Pringles can towers any different from writing for Presentation Manager?

What's The Secret?

The toughest problems usually turn out to be the ones involving other people. What we suspect seems to be true: Group dynamics may be the most important factor in software development group success. As Richard Cohen, speaking at SD '90, said "software development productivity is more strongly affected by people- and team-related issues than [by] any other variable under the management's control."

And teams are increasingly going to be where it's at. Programming in the large is — err — getting bigger. Ken Orr maintains that coding skill grows less and less important as the project gets bigger. "For people trained in good software engineering approaches, writing individual programs that aren't very large or complex is not a critical skill. On the other hand, the planning, architecture, requirements, and design of large suites of data files and programs (programming in the large) are critical skills [and] will become more so. The need for software engineers will remain acute, but increasingly these people will be writing systems, not programs."

If programming in the large is the task of the future, team programming is the paradigm of the future. Cohen nailed down what may be the essence of how successful teamwork feels: "A real team," he said, "is one in which the group feels common ownership of the problem and its solution." Many speakers at SD '90 talked about problems involved in team programming, managing software engineering projects, programming in the large. I didn't catch all their talks, but after the conference I sifted through the proceedings and my notes, and found that many of them dealt with the search for ways to achieve that sense of common ownership of the work.

It's an important search. Changes are afoot, SD '90 speaker Tim Twinam says, and the customer may well start calling more of the shots. The free ride of the Trappist technician may not end this year, but there is an inherent instability that will some day shake out.

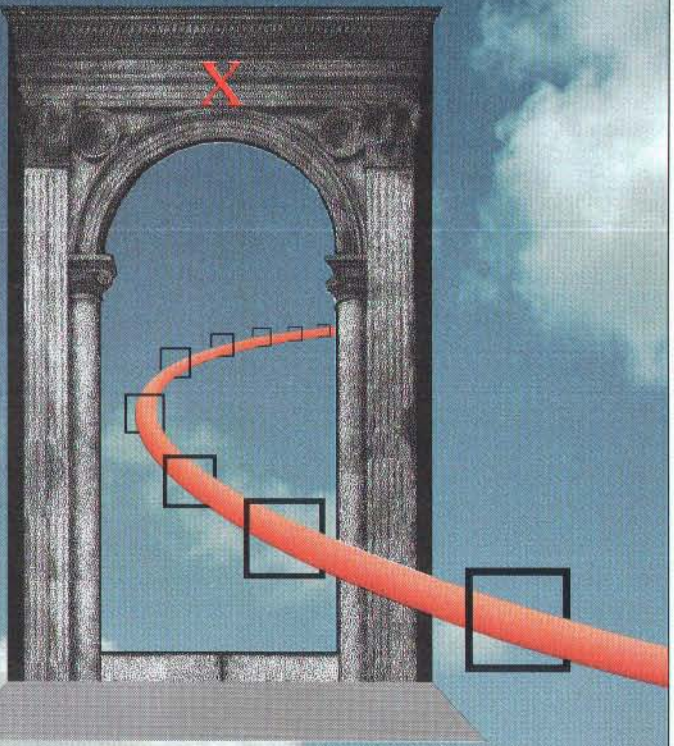
Vern Crandall and Larry Constantine have given some thought to the group structures conducive to good teamwork. Crandall, who worked at Novell, claims that fresh thinking is required in this area because most of the old answers, the software methodologies, including Orr's data-structured system development (DSSD), were developed for MIS development work, not for commercial software development. "We need a product-oriented approach," he said.

He lists some of the issues that are unique to or more important in commercial software development than in MIS:

- Ill-defined users (you can't walk down the hall and look over their shoulders to see what they're doing wrong);
- Programming to a moving target (the market changes during the development process);
- Multiple, complex models (the batch mode of early MIS is history; today a commercial product may have aspects of several models, including real-time operating system, embedded system, control software, distributed processing, communications, WAN, LAN, device driver, and database);
- Multiple industry standards and platforms;
- Many quality issues (reliability, installability, configurability, serviceability, usability, interoperability, performance, security, recoverability, and migration); and difficulties in customer education and support.

At least six groups of people must all work together well to make the project come off: Marketing, software development, software testing, software maintenance, documentation, and human factors. This is a complex, interrelated system of individuals with different skills and interests, all pushing to meet a common deadline that is invariably too tight. Crandall claims that flat and network management structures don't work in such an environment. He points out, furthermore, that software developers are creative people and that rules, regulations, and restrictions need to be kept to a minimum to avoid stifling their creativity. He hails the benefits of consensus management in maintaining the flexibility needed to let creative people solve problems together. But the flexibility and creativity can cause the developers to get off track, so a lot of supervision is necessary, as well as a lot of encouragement and direction.

XHIBITION 90



X AND BEYOND

What will computing be like in the 90's?
How will distributed computing affect business?
How do you take advantage of Open Systems?
How do you write software for the X Window System™?

May 21 to 24, San Jose, California

Call (617) 547-0510 for free poster and information packet

Xhibition 90 is sponsored by: Integrated Computer Solutions, Inc., 163 Harvard Street, Cambridge, MA 02139, Fax: (617) 547-0758, E-Mail *xhibit@ics.com.*

The X Window System is a trademark of the Massachusetts Institute of Technology.

CIRCLE NO. 284 ON READER SERVICE CARD

PAINT POWERFUL DATA-ENTRY SCREENS WITH POWER SCREEN™

Design screens and menus with PAINT™, placing input fields and messages precisely where you want them to appear in your final application. PAINT stores screen definitions in a single file — which can be modified without recompiling or relinking your application. Your program manages screen display and I/O through the POWER SCREEN Runtime Library. All essential features are supported, including:

◆ Block multi-field or single-field I/O ◆ Automatic range checking ◆ Configurable editing/menu key definitions ◆ Application context-sensitive, cross-referenced help ◆ Virtual/automatic scrolling screens within viewports ◆ Multiple, overlapping viewport display ◆ Plus much, much more!

The Runtime Library can be linked directly with Turbo Pascal applications, or installed as memory resident.

All this for just $149.00

Includes complete sample programs, a comprehensive reference manual, and the Norton Instant Access program and guides to assist you during program development. And, we offer a 30-day money back guarantee. Supports Microsoft C/QuickC, Turbo C, Turbo Pascal, QuickPascal and QuickBASIC.

Other powerful products from Blaise Computing:

C ASYNCH MANAGER™	**$189**
ASYNCH PLUS™	**$189**
View232™	**$189**
C TOOLS PLUS™	**$149**
Turbo C TOOLS™	**$149**
POWER TOOLS PLUS™	**$149**
POWER SEARCH™	**$149**

Call today for more information

(800) 333-8087 Fax (415) 540-1938

BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

Trademarks are property of their respective holders.

CIRCLE NO. 159 ON READER SERVICE CARD

He thinks that a hierarchical structure and an emphasis on accountability, challenge, and expectation is ideal.

How do you achieve this? Crandall has some suggestions: No individual team should be larger than six people. Products should be staggered, not sequentially released. Testing and maintenance are skills; hire testing and maintenance engineers, not testers, and pay them as much as the development engineers. Most of the existing models for a product life cycle are too long; you need to run in parallel as much as possible. In particular, software development, testing, documentation, and human factors (such as screen design) must run in parallel. This demands a lot of communication and a lot of freedom of movement, and that demands careful structuring of the teams.

People do care about quality and will work harder and better when they feel they are working on a product they can take pride in

Larry Constantine is a pioneer in the development of structured programming and structured design. He gave three talks at SD '90, and in one of them defined what he calls the "structured open team." It's apparently what Crandall is describing. "The structured open team uses formal structure to increase internal flexibility and adaptability while maintaining simple, external interfaces and behavior. Internally, it is structured to function as a tight-knit, closely integrated team of professional equals with clear differentiation of functions only as necessary for effective functioning." One of the key aspects of the structured open team is the default assignment of responsibility. Responsibilities can be shifted around as people's skills and knowledge and interests (and the changing demands of the job) require, but there is always a default assignment of responsibilities to ensure that nothing falls through the cracks.

One of the key steps in developing a software product is testing. Testing is naturally more complex in multiprogrammer projects, but Crandall turns up a surprising fact: Few schools offer any training in testing. Roetzheim, though, pointed out that testing is an important element of DoD specs. Maybe more programmers will be using DoD specs than I speculated earlier. Some companies today write to DoD specs in certain circumstances even when not doing Defense work.

But the most compelling point to come from these talks is that it is the human issues that are critical. Cohen listed some of the human traits that any software development team must deal with:

- People make mistakes
- People are often blind to their own errors
- People misunderstand each other
- People fixate
- People get overwhelmed by too many details
- People identify with their work
- People care about quality

The last point is a particularly good one for managers to keep in mind: People do care about quality and will work harder and better when they feel they are working on a product they can take pride in.

It was P.J. Plauger who sounded the contrarian note. There is always danger in accepting the common view uncritically, and there seems to be some sort of common view of good software management emerging (which is not to say that good software management is emerging). Plauger offered, for the stimulation of software management thinking, his contrarian list of software management heresies:

- Every software project must be just slightly out of control
- Your goal as a manager is to make software projects boring
- Your obligation to your programmers is to answer their telephone calls
- Your indispensable programmers are your greatest liability
- Teaching BAL programmers C++ is a waste of time
- Staying within budget is more important than making a profit
- Writing software must be fun, but not too much fun

Plauger justifies these heresies convincingly, but I think they are more useful if you are allowed to supply your own explanations.

DDJ

Vote for your favorite feature/article.
Circle Reader Service **No. 9.**

Prevent Serial Headaches

Greenleaf's CommLib 3.0 is Relief for C Programmers

Headaches:

If you're like most C programmers, communication issues are enough to give you a migraine. Interrupts are either impossible or require tricky assembler routines. File transfers are complicated by tricky and undocumented versions. Communications issues like modem control and handshaking using XON/XOFF or RTS/CTS simply aren't supported by your compiler. And lost data will send you for the aspirin bottle every time: your compiler just can't keep up with the ability of today's equipment to send torrents of data.

Fast Relief:

Greenleaf's C programming libraries have been solving headaches for some of the best programmers in the business for years [1]. CommLib 3.0 now ends communication programming problems faster. Interrupt routines handle every UART difficulty. File Transfers in **KERMIT, XMODEM, YMODEM**, and **ASCII** are just a function call away. Handshaking includes support for **RTS/CTS** and **XON/XOFF** at baud rates up to **115.2Kb** with up to **35 ports** on multiport boards. Modem control is easy with over 40 functions included. You can even run different baud rates and protocols on several ports at the same time. And you'll never worry about lost data again with CommLib's interrupt-level control.

Extras:

All Greenleaf function libraries include **source code** in C and in Assembler, **telephone support**, thorough manuals, and a subscription to the *Greenleaf Chronicles,* a C programmer's newsletter, at no extra charge. CommLib 3.0 includes the **PDQPlus Help Engine** and database. All major C compilers are supported.

End your communications headaches today. For information including free demo software or to place your order, call Greenleaf Software now:

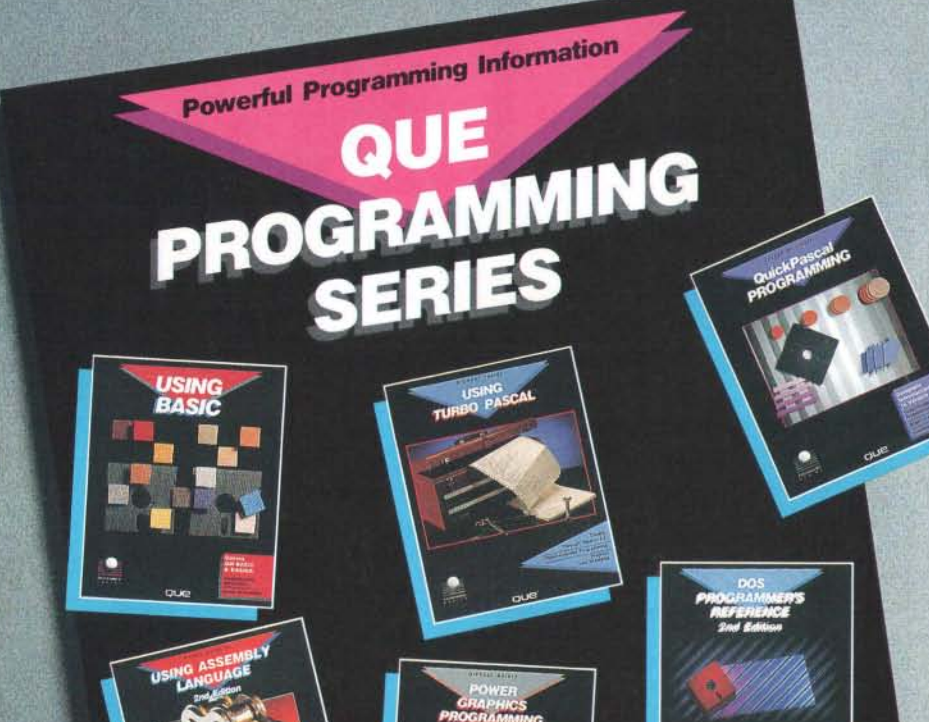
1-800-523-9830

FAX (214)248-7830

GREENLEAF Software©

[1]Greenleaf C libraries are used by programmers at AT&T Bell Labs, EDS, IBM, General Electric, General Motors, Northern Telecom, DuPont, Rockwell, Johnson Controls, Westinghouse, McDonnell Douglas, Kodak, Teledyne, Control Data, TI, and many others.

Program Your Success With Books From Que


Powerful Programming Information
QUE PROGRAMMING SERIES
USING BASIC
USING TURBO PASCAL
QuickPascal PROGRAMMING
USING ASSEMBLY LANGUAGE 2nd Edition
POWER GRAPHICS PROGRAMMING
DOS PROGRAMMER'S REFERENCE 2nd Edition
• Most Popular Environments
• Step-by-Step Guidelines
• Advanced Techniques
que
PROGRAMMING SERIES

No matter what programming environment you work in, Que books are your best resource. Packed with up-to-date information, skill building examples, and expert tips, Que provides the techniques you need to design powerful programs!

Using BASIC	$19.95
Using Assembly Language, 2nd Edition	$26.95
Using Turbo Pascal	$21.95
Power Graphics Programming	$24.95
QuickPascal Programming	$22.95
DOS Programmer's Reference, 2nd Edition	$27.95

Available in the U.S. and Canada at these fine stores:

BOOKS ETC.
Tempe AZ
602-967-1111

ASUA BOOKSTORE
Tucson AZ
602-621-2814

TOWER BOOKS SUNRISE
Citrus Heights CA
916-961-7802

OPAMP TECHNICAL BOOKS
Los Angeles CA
213-464-4322

USC BOOKSTORE
Los Angeles CA
213-743-2665

STACEY'S
Modesto CA
209-527-7120

TOWER BOOKS
Mountain View CA
415-941-7300

STANFORD BOOK STORE
Palo Alto CA
415-327-3680

STACEY'S
Palo Alto CA
415-326-0681

TOWER BOOK FLORIN
Sacramento CA
916-391-6121

TOWER BOOKS
Sacramento CA
916-481-6600

TOWER BOOK BROADWAY
Sacramento CA
916-444-6688

SAN DIEGO TECHNICAL BOOKS
San Diego CA
619-279-4990

STACEY'S-MARKET ST.
San Francisco CA
415-421-4687

STACEY'S-SACRAMENTO ST.
San Francisco CA
415-397-7935

COMPUTER LITERACY BOOKSHOPS
San Jose CA
408-435-1118

TOWER BOOKS
San Mateo CA
415-570-7444

MCKINZEY-WHITE BOOKS
Colorado Springs CO
719-590-1700

THE AURARIA BOOK CENTER
Denver CO
800-232-5280

REITER'S SCIENTIFIC & PROFESSIONAL BOOKS
Washington DC
202-223-3327

SIDNEY KRAMER BOOKS
Washington DC
202-293-2685

TROVER SHOPS
Washington DC
202-813-8006

NINTH STREET BOOKSHOP
Wilmington DE
302-652-3315

IT'S ACADEMIC BOOKSTORE
Orlando FL
407-658-4612

ENGINEER'S BOOKSTORE
Atlanta GA
800-635-5919

OXFORD BOOK STORES
Atlanta GA
800-476-3311

HONOLULU BOOKSHOPS
Honolulu HI
808-941-2274

57TH STREET BOOKS
Chicago IL
312-684-1300

CHAPTERS BOOKSHOP
Indianapolis IN
317-872-2665

PAGES AND PAGES-THE 1ST AMENDMENT BOOKSTORE
Louisville KY
502-425-2433

CAMPUS STORE-UNIVERSITY OF LOUISVILLE
Louisville KY
502-588-6985

CIRCLE NO. 431 ON READER SERVICE CARD

(continued from page 158)

HCR/C++, Version 2.2, is the first upgrade on this C++ compiler from **HCR Corporation**. HCR/C++ operates on most 386 and 486 Unix systems. This version includes class libraries that contain a comprehensive set of predefined objects that are compatible with the reusable object capabilities of C++. HCR/C++ comes with an enhanced release of dbXtra, the window-oriented debugger compatible with Berkeley's DBX.

HCR/C++ 2.2 includes the set of class libraries defined by AT&T and those in HCR/C++ 2.1, as well as the NIH (National Institute of Health) libraries, which provide classes for strings, linked lists, date and time conversions, indexed arrays, hash tables, regular expressions, and vector operations. The InterViews libraries that were developed at Stanford University and provide an interface to X Windows are also part of the package. The product should be available by now, and retails for $995. Version 2.0 customers can upgrade for $99. Reader service no. 23.
HCR Corporation
130 Bloor St. West, 10th Floor
Toronto, Ont. Canada M5S 1N5
416-922-1937

The bStrings Library for C, which adds dynamic string-handling capabilities to the C language, is available from **KBM Communications**. The bStrings Library provides dynamic strings without fragmenting memory. Over 130 string manipulation routines are provided, which duplicate most every string function available in Basic, as well as some not found in that language. The library supports functions that cut, copy, paste, clear, and overwrite whole strings or sections of strings, and will work with most screen management packages.

This library is available for Borland's Turbo C 2.0, Microsoft C 5.0/5.1 and QuickC 2.0. Both versions are provided with each order, and come with a 30 day money-back guarantee. The product sells for $89.95. Reader service no. 36.
KBM Communications, Inc.
2401 Lake Park Dr., Ste. 160
Atlanta, GA 30080
800-227-0303

A QuickBasic file indexing program, Index Manager, has been released by **CDP Consultants**. This product supposedly gives programmers the ability to create B+ tree files indexed within their QuickBasic programs. It allows random file access by full key, browsing through files by partial key, or sorting forward or backward. One external subroutine performs all of Index Manager's functions, and the programmer still retains full control over all data files, as only indexes are managed.

Indexes are created with a prefix B+ tree. The program is written in assembler language, and utilizes a large cache buffer for keeping important index records in memory. A demo version can be downloaded on CompuServe (GO MSSYS) on data library 1 or 2, called INDEXM.ARC, and GEnie (M 505) on data library 10, file 828. The program costs $59. Reader service no. 25.
CDP Consultants
1700 Circo del Cielo Dr.
El Cajon, CA 92020
619-440-6482

New run-time tools are now available from **Gold Hill Computers** for its development environments GoldWorks II and GCLISP Developer 3.1. The company claims that these products will produce applications that can be invoked directly from DOS or Microsoft Windows/286, that they will load as much as five times faster than applications loaded under the development environments, and that they will require much less memory.

GCLISP Runtime supports the delivery of GCLISP Developer 3.1 applications. The Lisp run-time configuration requires 1 Mbyte of extended memory, and so can deliver applications with less than 2 Mbytes of memory. Goldworks II/PC Runtime configuration requires 2 Mbytes of extended memory, with 4 Mbytes memory targeted for end-user machines, and is integrated with external programs such as Lotus 1-2-3, dBase III, and C. Gold Hill's StarterPak is $1000 for GCLISP 3.1 and $1500 for Goldworks II/PC. Reader service no. 27.
Gold Hill Computers, Inc.
25 Landsdowne St.
Cambridge, MA 02139
617-621-3300

DDJ

CONNECTING POINT COMPUTER CENTER OF LOUISVILLE
Louisville KY
502-456-5011

BARRON'S BOOKS INC.
Shreveport LA
318-868-5383

SOFTPRO
Burlington MA
617-273-2919

QUANTUM BOOKS
Cambridge Ma
617-494-5042

JOHNS HOPKINS UNIVERSITY BOOKSTORE
Baltimore MD
301-338-8000

MARYLAND BOOK EXCHANGE
College Park MD
301-927-2510

read all about it! BOOKSTORES
Omaha NE
402-341-7503

MCGRAW-HILL BOOKSTORE
Hightstown NJ
609-426-5748

COVER TO COVER
Plainsboro NJ
609-734-9233

CORNER BOOKSTORE
Albuquerque NM
505-266-2044

PAGE ONE, INC.
Albuquerque NM
800-521-4122

VILLAGE GREEN BOOKSTORE
Amherst NY
716-836-8960

VILLAGE GREEN BOOKSTORE
Buffalo NY
716-884-1200

VILLAGE GREEN BOOKSTORE
Greece NY
716-273-1600

MCGRAW-HILL BOOKSTORE
New York NY
212-512-4100

PAPYRUS BOOKSELLERS
New York NY
212-222-3350

CLASSIC BOOKSHOPS #61
New York NY
212-221-2252

COOPER SQUARE BOOKS
New York NY
212-533-6438

CLASSIC BOOKSHOP WORLD TRADE CENTER
New York NY
212-466-0668

VILLAGE GREEN BOOKSTORE
Rochester NY
716-461-5380

INTIMATE BOOKSHOPS
North Carolina
800-TELE-BUY

BOOKSELLERS AT PAVILLIAN MALL
Beachwood OH
216-831-5035

UNIVERSITY OF CINCINNATI BOOKSTORE
Cincinnati OH
513-556-1801

LAZARUS BOOK DEPT. DOWNTOWN
Cincinnati OH

READ-MORE BOOK STORE
Columbus OH

LAZARUS BOOK DEPT. DOWNTOWN
Columbus OH

MICRO CENTER
Columbus, OH
Marietta GA

BOOKSELLERS AT ROCKY RIVER
Rocky River OH
216-333-7828

PORTLAND STAFF BOOKSTORE
Portland OR
503-226-2631

GLOBAL BOOKS
Feasterville PA
215-357-8258

GENE'S BOOKS
King Of Prussia PA
215-265-6210

UNIVERSITY OF PA BOOKSTORE
Philadelphia PA
215-898-4900

BOOK CENTER-UNIVERSITY OF PITTSBURGH
Pittsburgh PA
412-648-1452

CHESTER COUNTY BOOK CO.
West Chester PA
215-696-1661

Taylors Technical Books
Dallas TX
214-357-1700

PRNT Bookstore at Infomart
Dallas TX
214-746-3625

MAXWELL BOOKS
Desoto TX
214-224-9127

MAJORS SCIENTIFIC BOOKS
Houston TX
713-522-1361

BROWN BOOK SHOP
Houston TX
800-423-1825

BARRON'S BOOKS INC.
Longview TX
214-663-2060

SAM WELLER'S ZION BOOKSTORE
Salt Lake City UT
801-328-2586

TOWER BOOKS BELLEVUE
Bellevue WA
206-451-1110

UNIVERSITY BOOKS
Seattle WA
206-634-3400

THE ELLIOTT BAY BOOK COMPANY
Seattle WA
206-MAIN4-6600

TOWER BOOKS
Seattle WA
206-283-6333

LE CAMELOT COMPUTER
Montreal Quebec
CANADA

ODYSSEY COMPUTER & BUSINESS BOOKS
Ottawa Ontario
CANADA

WORLD'S BIGGEST BOOKSTORE
Toronto Ontario
CANADA

SILICONNECTIONS BOOKS LTD.
Vancouver B.C.
CANADA

que

New
Lower Price
and **Free**
PRO-C Work Bench

"Mary had a 4GL whose performance was very slow and everywhere that Mary went the run-times had to go."

PRO-C
The C Source Code Applications Generator

Accelerate Your Productivity

Pro-C gives you the greatest gift in the computer world – time. You get high quality, fully commented, error free C source code in a fraction of the time it would take to write it by hand.

Pro-C has always **saved** you time and that **saved** you **money**. Now you **save** even more. Vestronix has dropped the price of Pro-C from $675 US to $399.00! And Pro-C Work Bench, the C source code libraries that let you quickly customize your application, is absolutely FREE! A great program at great savings.

Pro-C will increase your ability to create programs quickly by generating the source code for menus, reports, screens, windows, and multi-file batch updates. Textbook quality C code is written by us, while the elegant system solutions and exciting new algorithms are created by you.

Pro-C looks and feels like a 4GL, but it's not. You can do prototyping, layout, design and generation of applications without learning a proprietary language, needing massive amounts of memory, or ending up with slow running programs. Best of all Pro-C doesn't require any run-time licenses. Finally, a company that treats you like royalty instead of forcing you to pay them.

Pro-C – the programming partner that does the boring, repetitive coding without complaint, pays for itself every time you use it, and doesn't argue with your obviously brilliant program designs.

With PRO-C, everybody's a winner.
CIRCLE NO. 104 ON READER SERVICE CARD

PRO-C Order yours to-day. Call 1-800-265-2682 VESTRONIX

CSORT: A Saga of a Sort

Al Stevens

Those of you with mainframe experience know that a common and indispensable utility program in that environment is the file sort program. Many batch applications involve the generation of reports taken from data files that you maintain in sequences other than the sequence needed for the report at hand. I remember the two- and three-way tape sorts on the IBM 1401 from as far back as 1961. It would have been unthinkable to try to design a system without one.

The Mainframe Tape Sort

Here's how such a sort works in the typical tape-drive configuration of yore. You need at least four tape drives to run a two-way sort. The unsorted file starts out on the first tape drive, and the other three have scratch reels of tape. A file of parameters (usually on a control card) tells the sort program the size of the file's records and the location, length, and sequence (ascending or descending) of each of the fields to be sorted.

Pass One, the Input Pass — The sort program reads as many records into memory as the CPU can hold and sorts them in an array. Then it writes the sorted block of records to the third tape drive. Next, it reads and sorts another block of records and writes that block to the fourth drive, then the next block to the third, and so on until all the input records have been read. Drives 3 and 4 now each contain multiple blocks of individually sorted records, and this is the end of the first pass.

During the first pass, the block size is restricted to the amount of memory that is available. Block sizes double during subsequent passes because each block is in sequence, and the program reads them a record at a time to perform the merge.

Pass Two, the Merge-down Pass — The computer operator replaces the input tape on drive 1 with a fourth scratch tape, and the sort program merges the first block from drive 3 with the first block from drive 4, writing the merged block onto drive 1. Then it would merge the next blocks from drives 3 and 4 onto drive 2. This flip-flop continues until all the blocks from drives 3 and 4 are merged into drives 1 and 2, which together now contain half as many blocks as 3 and 4. (The old-timers among my readers are beginning to get bored.)

Pass Three, the Output Pass — The merges continue back and forth with each pass reducing the number of ordered blocks by half and doubling the size of each block until at the final pass there is only one block on either drive 1 or 3, and this tape contains the sorted file, which can be read by the application.

Sordid Data on Micros

When I began working with microcomputers in the mid-seventies I looked for the standard sort utility program that I thought would come with operating systems, and I found none. The first significant application I wrote for an IMSAI 8080 needed file sorting, and a search turned up a CP/M program called "QSORT," but I was surprised to learn that it was difficult to find what had always been an indispensable utility program.

With the PC and MS-DOS came the SORT filter program, but it has three serious limitations: It can only sort as many records as will fit into 64K, it can sort only one field of the file's records, and, being a filter, it works only with text files.

The In-line Sort

So much for the file sort utility program. Now let's consider the in-line sort feature. Anyone who has programmed in many computer languages has accumulated some favorite features about each one. The ideal personal language would have all those features wrapped into one. Of course, when you try to build a language with everyone's favorite features, you get a committee-designed language — you get ADA, perhaps. The perfect personal language would be so extensible that each of us could design his own syntax and data types, plugging in the features we treasure and leaving out the ones we do not. The flaw in that approach is that no one could read anyone else's code, and so, instead of each of us building a personal ideal language, we ebb to the committee approach, standards emerge, and we strive to conform.

As a full-time C programmer and writer, I often think of language features in terms of the features I liked about languages past that C does not have. For example, the string functions of Basic are missing in C. Cobol's MOVE CORRESPONDING would be handy in C where the C structure assignment would assign only those members that have the same names. You can add both of these features to C by building appropriate C++ classes if you are using the C++ extensions to C, and so the dubious realm of the customized language looms again in the near future.

There is one feature I liked about Cobol that you can build in traditional C without extending the language other than with a few functions. That feature is the SORT verb and it is relevant to this month's rambling.

A Cobol program can pass records to the SORT verb one at time and then later retrieve those records in the sorted sequence. There are several advantages to this technique. One is that the program does not need to prepare an unsorted file, exit to a sort utility program, and then execute a third program to process the sorted file. You can form each unsorted record from one or more sources just prior to sending it off to be sorted, and you can transform the sorted records into another format before doing anything with them. No intermediate file of uniquely formatted records is involved.

The C *qsort* function is similar to this approach except that it sorts an in-memory array, and so all your data records must fit into memory. A true in-line sort will accept as many records, one at a time, as you want to give it, and will return those records to you, one at a time, in sorted order.

So now we have defined two requirements missing from the C language and the microprocessor operating system environment. Why, after ten solid years of micro use, are these features not standard? Obviously, they are not widely needed for some reason, and that reason is, I guess, that the personal, inexpensive nature of the PC has redefined the way we design systems. Most PC programs today are interactive. If they involve multiple sequences of data files, they use indexed database managers or higher-level DBMS languages that deal with sorting in their own ways. OK, so most of the time we do not need a sort program or an in-line sort. But what about the few times when we do? Nothing else will do.

About five years ago, I tackled this problem by writing a C language in-line *sort* function for some programs for a video tape store. Later I used it as the sort engine for a stand-alone sort program that sorts disk files. I used pre-ANSI Aztec C for the IBM-PC and published the code in a book (*C Development Tools for the IBM PC*, Brady Books) in 1986. Since then I have reused that program many times in circumstances where I might have chosen a more accessible but less appropriate solution had I not already had this little sort program. Now that the ANSI standard definition of C is in place, it's time to update that program, so this month we promote it to the standard and publish it here.

CSORT

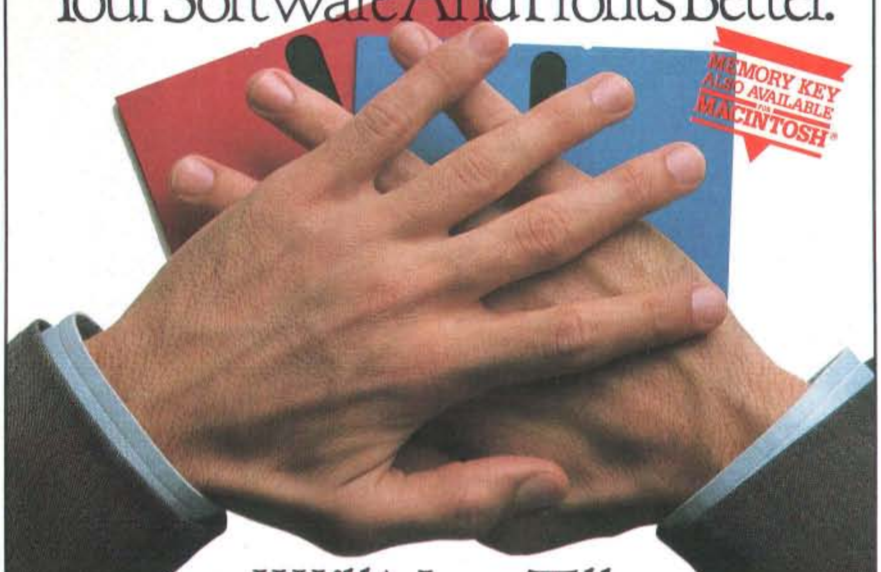
What follows is CSORT, a sorting facility that you can use from within your programs or from the command line. Although I developed it to use in CP/M systems and later in MS-DOS, it is not wired to either of those environments and should easily port to another platform where a standard C compiler is available.

The CSORT project has two parts, the in-line sort and the file sort program. To use the in-line sort in a C program, you describe the characteristics of the records to be sorted and send them, one at a time, off to the sort process. After you have sent the last of the records, you send a NULL and go about your business. When you are ready for the records in the sorted sequence, you call the sort program and it returns the records, one for each call. After it has returned the last of the sorted records, it returns a NULL. Your program is essentially the polar ends of a three-phase pipe.

You and CSORT pass records back and forth with pointers. CSORT makes a copy of the record you pass it, so you can safely reuse the space the record occupied. CSORT expects you to do the same because it might reuse the record space that is pointed to to be a previous pointer it returned.

CSORT requires that you sort fixed-length records, but the records themselves do not need to be in text format. The sort comparison uses the *strnicmp* function to compare fields, so they do not need to be null-terminated.

Here's How We Protect Your Software And Profits Better.



We'll Never Tell.

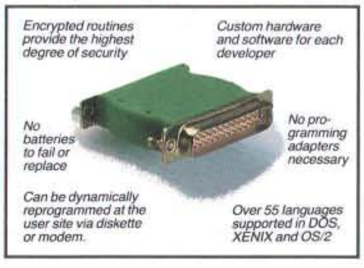
Because our key-interrogation routines are *encrypted*, and our hardware is custom-wired to distinguish each of our clients' keys, our clients have the highest degree of security available. Unlike other manufacturers, our routines assume responsibility for all hardware, software and timing issues. And this means your engineering time and money won't be wasted reinventing protection schemes.

We offer two high security products for copy control: the KEY and the MEMORY KEY.™

Our protection devices can also be used for serialization techniques, software leasing, modular software management, creative revenue collection, demo control and a path for future upgrades.

The information stored in the MEMORY KEY can be conveniently reprogrammed by your application software or at the end user's site via software disk or modem.

All our products attach conveniently to the printer port, are transparent and allow for unlimited back-ups.


Encrypted routines provide the highest degree of security
Custom hardware and software for each developer
No batteries to fail or replace
No programming adapters necessary
Can be dynamically reprogrammed at the user site via diskette or modem.
Over 55 languages supported in DOS, XENIX and OS/2

For serious software protection, call now. And start protecting your profits. Hands down, we're better.

PROTECH MARKETING, INC.

9600-J Southern Pines Blvd. Charlotte, NC 28217
Tel: 704-523-9500 Fax: 704-523-7651
Hours: Mon.-Thurs.: 8:30-7:00, Fri.: 8:30-5:30 ET
For a demonstration package or additional information, please write or call:
Se Habla Español **1-800-843-0413**

Macintosh is a registered trademark of Apple Computer, Inc.

CIRCLE NO. 210 ON READER SERVICE CARD

Now Supports DLL
WINDOWS

If You Want To Talk Fast DBMS Call 1-800-db-RAIMA And Start Screaming

You'll be screaming, all right. db_VISTA III from Raima combines the flexibility of a relational DBMS and the lightning speed of the network database model.

C db_VISTA III is written for C Programmers. Source code available. The interactive database utilities and outstanding documentation make db_VISTA III easy to learn. All applications are portable to VMS, UNIX, OS/2, MS-DOS, even Macintosh. No royalties.

db_VISTA III is *Fast*. Using benchmarks originated at PC Tech Journal Laboratories, db_VISTA III measured *3 to 12 times faster than the average relational database!* Call us and we'll send you the results.

db_VISTA III Database Management System

Features	Yes	No
db_VISTA 3.1 High Performance DBMS:		
Single and Multi-User available	✓	
Relational B-tree Indexing	✓	
Network Database Model	✓	
Multiple database access	✓	
Referential integrity	✓	
Automatic recovery	✓	
Record & File locking	✓	
RAM resident		✓
db_QUERY 2.1 SQL-based Query:		
Relational Query & Report Writer	✓	
db_REVISE 1.0 DBMS Restructure Program:		
Total database redesign/restructuring	✓	
C Compilers*: Most supported	✓	
C++ compatible. Supports **PRO-C**	✓	
Operating Systems*: VMS, ULTRIX,	✓	
UNIX, BSD, SunOS, XENIX, QNX,	✓	
MS-DOS, MS Windows, and Macintosh.	✓	
OS/2 compatible.	✓	
LANs*: 3COM, Novell, Banyan,	✓	
AppleShare and more ...		
WKS Library:		
Read & Write WKS, WK1 & DBF files	✓	
SOURCE CODE AVAILABLE:	✓	
ROYALTIES: (Absolutely not!)		✓

*Other environments are supported; call for complete list.

Relational and Network Model Technology for Programming Flexibility. Retrieve a record fast using the relational keyed access method and all related records can be immediately available using the network database model. You decide how to combine these for best application performance.

SQL **SQL Support** with SQL-based db_QUERY, db_VISTA III's relational query and report writer.

db_VISTA Puts You in Some Fast Company. Thousands of C programmers in over 50 countries worldwide use db_VISTA III, including APPLE, ARCO, AT&T, EDS, Federal Express, Hewlett-Packard, IBM, NASA...

Don't wait. Call Raima for more information about how you can build *screaming-fast* applications!

db_VISTA III™
Database Management System

CIRCLE NO. 420 ON READER SERVICE CARD

RAIMA™ CORPORATION

No Matter What Your Operating System-We've Got A Number For You!

1-800-db-RAIMA
(1-800-327-2462)

Raima Corporation 3245 146th Place S.E., Bellevue, WA 98007 USA	**(206)747-5570** **FAX:** (206)747-1991 **Telex:** 6503018237 MCI UW

International Distributors:

U.K.: (0992) 500919	**Austria:** 02243-81861
Germany: 07127/5244	**Australia:** (02) 419 7177
Netherlands: (02159)46 814	**Japan:** (03)473 7432
Switzerland: (01)725 0410	**Taiwan:** (02)511 3277
Sweden: (013)124780	**Mexico:** (83) 57 35 94
Italy: 045/584711	**Argentina:** 1 313 5371
Norway: 244 88 55	**Chile:** 2 696-4308
Denmark: (2)887249	**Uruguay:** 2 92 0959
U.S.S.R.: (812) 292-19-65, (0132) 35-99-08	**Central Am.:** (506) 28 07 64
	Brazil: 011 532 1689

DD04

(continued from page 128)

The CSORT API

Following is the application program interface for CSORT. Listing One, page 144, is CSORT.H, which contains several control values for the sort. The NOFLDS global value specifies the maximum number of fields that can be involved in the sort of the record. The MOSTMEM and LEASTMEM values control the appropriation of a buffer from the heap for sorting. MOSTMEM is the amount you try for, and LEASTMEM is the least amount you will accept. I have them set to work within a small-model program on the PC.

Here are the prototypes and descriptions of the CSORT API functions.

int init_sort(struct s_prm *prms); To sort records, you must describe them. The *s_prm* structure contains the definition of the records to be sorted. It is defined in CSORT.H. You will declare the structure and initialize it as follows. Assign the record length to the *rc_len* member. The *s_fld* array of substructures will contain entries for each of the fields to be sorted. The *s_fld.f_pos* member contains the record position for the field. This value is relative to one. If there are fewer than NOFLDS fields, the terminal entry in this array will contain a zero value for this member. The *s_fld.f_len* member is the length of the field. The *s_fld.ad* member is the character "a" if the field is to be collated in ascending sequence and "d" if it is to be collated in descending sequence. The first field in the array is the major sort field; the last is the minor; all others are intermediate. So, if you need records in, for example, division number, department number, and employee number sequence, you will define the fields in that order in the array. With the array properly initialized, call the *init_sort* function. If the sort may proceed — if enough memory is available — the function returns zero. Otherwise it returns –1.

void sort(char *rcd); With the sort program properly initialized through *init_sort*, you may send records to be sorted by calling the *sort* function once for each record. Pass a pointer to the record, and the *sort* function will make a copy of the record. After you have passed the last record, pass a NULL pointer. This tells the sort function to finish the sort and prepare to return sorted records.

char *sort_op(void); To retrieve sorted records, call the *sort_op* function. Each call to it will return a pointer to the next record in the sorted sequence. After the last record comes back, the *sort_op* returns a NULL pointer.

void sort_stats(void); This function displays some of the values used in the sort function.

The FILESORT Program

Listing Two, page 144, is FILESORT.C, a program that uses the CSORT API to implement a stand-alone file sort program. You run it by entering the filename, record length, and field parameters on the command line. It uses the API in the manner just described.

CSORT Internals

Listing Three, page 144, is CSORT.C, which contains the in-line sort functions. The first function to discuss is *init_sort*, which you call to initiate sorting. It calls *appr_mem* to allocate a block of memory in which to sort, initializes the sorting parameters, and returns.

The *sort* function accepts records to sort from the calling application. At first it simply copies them into the buffer, one after the other. When the buffer is full, the *sort* function calls the standard *qsort* function to sort the records in the buffer. This becomes a block of sorted records, also called here a "sequence." The *dumpbuff* function writes them to the sort work file. CSORT uses only

ASPIRIN

ONLY $250!

The C-Programmer's Pain Reliever

- **C Programs giving you headaches?**
- **Tired of having to rewrite code to fit into your latest programs?**
- **Wish you could avoid the hassle of trying to use four different libraries?**

We all know that the standard C library just doesn't cut it in today's market of fast, user-friendly programs. We also know that trying to link together 3 or 4 different add-on libraries is often more of a headache than it is worth.

There is a solution. **ASPIRIN,** the new package from *Arrowhead Software,* eliminates those aches and pains at a price that won't hurt your wallet!

ASPIRIN is not just another forms/windows package. You'll have all these capabilities at your fingertips, plus many other time-saving features:

* ISAM database package
* Text manipulation
* C-code generator
* Linked - lists
* Mouse handling
* Date and time functions
* Windows
* Forms
* Menus
* Utility programs
* Step-by-step user's guide
* 300 page technical guide

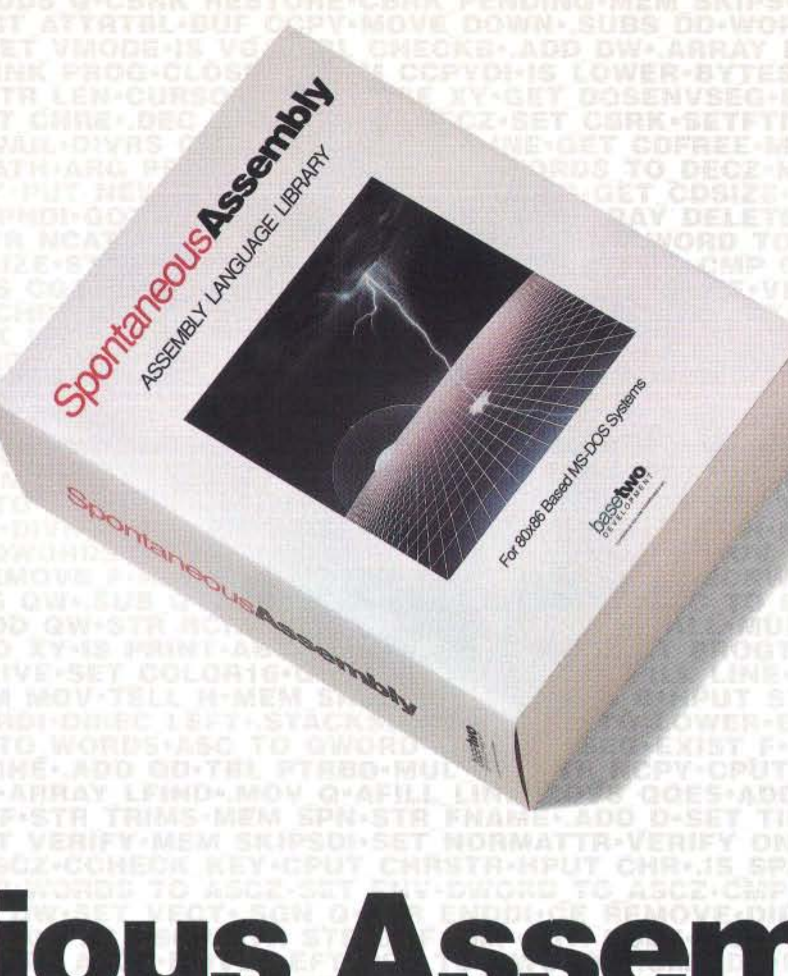
How ASPIRIN Stacks Up

	Vitamin C	C-Scape	Window	Vermont	Greenleaf	C Worthy	ASPIRIN
Windows	Yes	Yes	Yes	Yes	Yes	Yes	YES
Fo:ms	Yes	Yes	Yes	Yes	Yes	Yes	YES
Menus	Yes	Yes	Yes	Yes	Yes	Yes	YES
Code Generator	No	Yes	No	No	No	No	YES
Data Structures	No	No	No	No	No	No	YES
ISAM	No	No	No	No	No	No	YES
Utilities	Yes	No	Yes	Yes	Yes	Yes	YES

ARROWHEAD SOFTWARE

7500 W. Mississippi, Ste. 201
Lakewood, CO 80226
(303) 922- 1300
FAX (303) 922-1324

CIRCLE NO. 312 ON READER SERVICE CARD



Serious Assembly.

Nobody really takes assembly language seriously. Nobody, that is, except Lotus, WordPerfect, Novell, and everyone else who needs the tightest, fastest code possible.

But assembly is tough . . . unless you have plenty of ready-to-use assembly language routines at your fingertips. That's why industry leaders spend a lot of time and money creating their own, proprietary assembly language libraries.

You could spend your valuable development time building a library; or you can use **Spontaneous Assembly**, the complete **assembly language library** specifically designed for serious assembly language development.

Over 600 functions and macros.

- String manipulation • Memory manipulation
- Near/far/relative heap management
- Doubleword/quadword integer math
- Array management, sorting, and searching
- Character/numeric/string conversion • Buffer management • Date and time manipulation
- File and directory management • File I/O
- Keyboard I/O • Program/environment control
- Critical error management • Screen I/O with windowing—and more. All carefully documented, tested, and ready to use.

Tight, fast code. Fast.

Every routine is hand-coded and hand-optimized (in assembly, of course). And a consistent, register-oriented parameter-passing convention makes these routines remarkably easy to use. With Spontaneous Assembly, coding in assembly is as fast as coding in a high-level language—without the overhead.

Powerful memory model support.

Spontaneous Assembly unleashes the full power of memory models in assembly language. It supports all Microsoft/Borland standard memory models as well as custom models and mixed-model programming. And it gives you complete control over segment/group names and attributes.

Extensive documentation.

A complete reference manual describes every function, macro, and variable in detail. Step-by-step instructions and technical notes explain memory models, library customization, integration with C, and much more.

Spontaneous Assembly.

Serious assembly language programming doesn't need to be hard any more. Now it can be Spontaneous.

See for yourself. Call now to take advantage of our introductory price. Then use Spontaneous Assembly for 60 days. If you're not 100% satisfied, send it back for a full refund!

Introductory Price.

$199 Plus Shipping & Handling.

Full source code included. No royalties.

60 day money-back guarantee.

Visa • MasterCard • Discover

Call now!

1-800-ASSEMBLY

Orders • Information • Support

basetwo DEVELOPMENT

For 80x86–based systems running DOS 2.0 or later. MASM 5.1 or TASM 1.0 recommended.

Base Two Development • A Division of Acclaim Technologies, Inc. • 11 East 200 North • Orem, Utah 84057 • (801) 222-9500

Spontaneous Assembly and Base Two Development are trademarks of Acclaim Technologies, Inc.

(continued from page 130)
one sort work file, unlike the tape sorts we discussed earlier. Because this program uses disk files, and because direct record addressing is possible, we do not need to maintain multiple, serial devices for the blocks of sorted records after the fashion of a streamer tape device.

When the calling application sends a NULL pointer to the *sort* function, it sorts the final sequence, writes it to the work file, and calls *prep_merge* to prepare for when the caller wants sorted records. The merge divides the sort work buffer into many miniature buffers, one for each sequence that was generated by the *sort* function. These buffers contain two or, usually, more records. If and while the number of sequences is high enough to prevent the buffer from being segmented this way, *prep_merge* uses the *merge* function to merge groups of two sequences into one, halving the number of sequences and doubling their sizes.

When the number of sequences is low enough that each one can contribute at least two records in the sort buffer, the *merge* and *prep_merge* functions are done.

The calling application calls *sort_op* to get sorted records. The *sort_op* function looks at the first record in each buffer segment to find the lowest record. It bumps that segment's record pointer and returns a pointer to the record it found. When it exhausts a buffer segment, it reads more records from the associated sequence on the sort work file. When all the segments are empty, the application has read the last sorted record, and the sort program signals that it is done by returning a NULL pointer.

TopSpeed C

Last year I reported that I had seen a demo of the announced TopSpeed C compiler from Jensen & Partners International. The product is shipping now, and is a full-featured ANSI-conforming compiler with integrated editor, compiler, linker, and debugger. I used it to upgrade CSORT to the ANSI specification and can offer these first impressions. Be advised that I have not yet used TSC to build a significant system from scratch, so these judgments are preliminary. I think you will like this product. After reading and following the installation procedures, I used the books only once during the project, and that was to learn the format for the MAKE project file. Everything else is neatly supported with comprehensive help windows and intuitive menus.

TSC has a number of features that I haven't used yet but surely will. They have built a DOS equivalent of the OS/2 dynamic link library (DLL). You can build programs that link to their libraries at run time. There is a postmortem debugger, an assembler and disassembler, a program profiler, and support for DOS Windows and OS/2 Presentation Manager program development.

TSC has a few things I'd change. Their version of the *interrupt* function type has a severe disability. You cannot call an interrupt function directly or through a function pointer. This prevents you from chaining intercepted interrupts and renders TSC ineffectual as a TSR development compiler — unless you want to compensate by using assembly language.

Although TSC has the _AX, _BX, etc. pseudoregisters of Turbo C, they do not support _FLAGS, which makes their use in interrupt service routine programming even more difficult.

The tables of contents and indexes in most of the documents do not agree with the actual page numbers. This is an unacceptable lapse in quality control for any documentation effort. I forgive that lapse only because the on-

TAKE CONTROL

CURSES and FORMATION.
The Key to UNIX, DOS, OS/2 and VMS Window Portability.

CURSES: The library for C developers who want uncluttered portability and power in their screen and keyboard interfaces. CURSES' device independence and fully optimized output allows your application to have only one set of C source files, the key to portability.

CURSES/PC New Version! Full Unix System V.3.2 CURSES including color support. Supports EGA 43 and VGA 50 line modes, and 101 keyboard.
DOS-OS/2 Source& Binary Bundled -$349
DOS-OS/2 Binary Bundled -$159

CURSES/VMS supports all terminals defined by VMS TERMTABLE. Single user license - **$799**

FAST with source, the CURSES Forms Manager - **FREE** with CURSES.

FORMATION: Dialog boxes, exploding viewports and sophisticated menus - normally found only under graphics environments - are all part of this grown-up human interface tool. And, of course, portability to environments supporting CURSES.
DOS-OS/2-UNIX-VMS Source & Binary Bundled -$399
DOS-OS/2 Binary Bundled -$159

Take Control. "To get the desired terminal display compatibility, I bought from Aspen Scientific." –*DEC Professional*
Do the same, call now!

ASPEN SCIENTIFIC
P.O. BOX 72 WHEAT RIDGE, COLORADO 80034-0072
303-423-8088
FAX 303-420-6522

Overseas Distributors: W. Germany: Kickstein Software 0821-81 46 66 • Holland: Lemax 02968-4210 Radix 02014 24
Trademarks: - OS/2: Microsoft; UNIX: AT&T; VMS: Digital Equip. Corp.

line documentation is so good.

TSC tends to leave my cursor other than the way it found it, and it leaves all manner of what appears to be temporary work files scattered about my source directory. These quirks are annoying at best.

One last gripe, then I'll relax. I rejoiced when I saw the WATCH utility. It is a really neat TSR, tossed in for good measure, that you can use to monitor the calls your program makes to DOS, an essential tool for systems programmers. I do a lot of programming in the Novell NetWare local area network environment, and NetWare API calls are supersets of the DOS INT 0 x 21. To my dismay, I learned that WATCH only watches calls that you select from its list of known DOS calls, and there seems to be no way to tell it to watch the Novell superset list. Rats. Another idea well conceived, poorly built.

If it seems that I am overly critical it is because these are the things I found that I think you should know about. There may be others, but, as I said, my use of TSC has been limited so far. All that aside, I like the product and do not hesitate to recommend it. It is in great shape for the first version of any software product and will surely improve with newer versions. The PC programmer has a dilemma. There are several excellent compilers to choose from. By now, there will have been benchmarks and reviews and, I am sure, you still will not know what to do.

(The title of this month's column is a paraphrase of that of the Bill Mauldin book, *A Sort of a Saga.* His book has nothing to do with C or programming, but Mauldin's cartoons and writings and that book in particular are timeless and were major influences in my young life.)

Availability

All source code is available on a single disk and online. To order the disk, send $14.95 (Calif. residents add sales tax) to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call 800-356-2002 (from inside Calif.) or 800-533-4372 (from outside Calif.). Please specify the issue number and format (MS-DOS, Macintosh, Kaypro). Source code is also available online through the *DDJ* Forum on CompuServe (type GO DDJ). The *DDJ* Listing Service (603-882-1599) supports 300/1200/2400 baud, 8-data bits, no parity, 1-stop bit. Press SPACEBAR when the system answers, type: listings (lowercase) at the log-in prompt.

DDJ

(Listings begin on page 144.)

Vote for your favorite feature/article.
Circle Reader Service **No. 10.**

Commenting Disassembler!

SOURCER™ 486

- **SEE HOW PROGRAMS WORK**
- **EASILY MODIFY PROGRAMS**

SOURCER™ creates detailed commented source code and listings from memory and executable files. Built in data analyzer and simulator resolves data across multiple segments and provides detailed comments on interrupts and subfunctions, I/O ports and much more. Determines necessary assembler directives for reassembly. Includes a definition file facility to include your own remarks and descriptive labels, force data types, and more. Complete support for 8088/87 through 80386/387, 80486, and V20/V30 instruction sets. We welcome comparisons with any other product, because no product comes close to the ease of use and output clarity of SOURCER.

Sourcer is the best disassembler we've ever seen!
—PC Magazine, January 17, 1989, page 101

SAMPLE OUTPUT

Fully automatic

Program header

Assembler directives

Determines data areas and type

Detailed comments

Simulator follows segment changes

80386 and 80486 support

Easy to read format

```
        resetprn.lst    ResetPRN v1.02                   Sourcer Listing   18-Sep-89   1:42 pm      Page 1

                                  PAGE  60,132
                                  ;
                                  ;                      RESETPRN
                                  ;
                                  ;          Created:   24-Aug-89
                                  ;          Version:   1.02
                                  ;          Passes:    8          Analysis Flags on: H
                                  ;

                                  .386c

         = 0008                   @prn_port_1   equ     8                     ; (0040:0008=378h)

                                  ;------------------------------------------------------------ seg_a -----
                                  seg_a         segment para use16 public
                                                assume cs:seg_a, ds:seg_a, ss:stack_seg_b

                                  resetprn      proc    far
658E:0000                         start:
658E:0000  EB 23                                jmp     short loc_1
658E:0002  52 65 73 65 74 50                    db      'ResetPRN v1.02', 00h
658E:0008  52 4E 20 76 31 2E
658E:000E  30 31 00
658E:0011  0040                   data_2        dw      40h
658E:0013  0D 0A 52 65 73 65      data_3        db      0Dh, 0Ah, 'Reset Printer? $'
658E:0019  74 20 50 72 69 6E
658E:001F  74 65 72 3F 20 24
658E:0025                         loc_1:
658E:0025  0E                                   push    cs
658E:0026  1F                                   pop     ds
658E:0027 .BA 0013                              mov     dx,offset data_3    ; (658E:0013=0Dh)
658E:002A  B4 09                                mov     ah,9
658E:002C  CD 21                                int     21h                 ; DOS Services  ah=function 09h
                                                                            ;  display char string at ds:dx
658E:002E  B4 01                                mov     ah,1
658E:0030  CD 21                                int     21h                 ; DOS Services  ah=function 01h
                                                                            ;  get keybd char al, with echo
658E:0032  3C 79                                cmp     al,79h              ; 'y'
658E:0034  75 1A                                jne     short loc_3         ; Jump if not equal
658E:0036  8E 1E 0011                           mov     ds,data_2           ; (658E:0011=40h)
658E:003A  8B 16 0008                           mov     dx,ds:@prn_port_1   ; (0040:0008=378h)
658E:003E  83 C2 02                             add     dx,2
658E:0041  B0 08                                mov     al,8
658E:0043  EE                                   out     dx,al               ; port 37Ah, printer-2 control
                                                                            ;  al = 8, initialize printer
658E:0044  66| B9 00020000                      mov     ecx,20000h
658E:004A                         locloop_2:
658E:004A  67| E2 FD                            loopd   locloop_2           ; Loop if ecx > 0
658E:004D  B0 0C                                mov     al,0Ch
658E:004F  EE                                   out     dx,al               ; port 37Ah, printer-2 control
                                                                            ;  al = 0Ch, init & strobe off
658E:0050                         loc_3:
658E:0050  B4 4C                                mov     ah,4Ch              ; 'L'
658E:0052  CD 21                                int     21h                 ; DOS Services  ah=function 4Ch
                                                                            ;  terminate with al=return code
                                  resetprn      endp
                                  seg_a         ends

                                  ;------------------------------------------------------------ stack_seg_b ----
                                  stack_seg_b   segment para use16 stack
6593:0000  00C0[FF]                             db      192 dup (0FFh)
                                  stack_seg_b   ends

                                                end     start
```

(Source code output and inline cross reference can also be selected)

BIOS SOURCE

- **CHANGE AND ADD FEATURES**
- **CLARIFY INTERFACES**

for PS/2, AT, XT, PC, and Clones

The BIOS Pre-Processor™ with SOURCER provides the first means to obtain accurate legal source listings for any BIOS! Identifies entry points with full explanations. Resolves PS/2's multiple jumps for improved clarity. Provides highly descriptive labels such as "video_mode" and much more. Fully automatic.

SOURCER Commenting disassembler	$99.95	**ASMtool™** Assembly source analyzer and flowcharter	$89.95
SOURCER with BIOS Pre-Processor	139.95	**View-It™** High speed zoomable file viewer	69.95
UNPACKER™ Unpack packed EXE file	39.95	**ASM ProPak™** Sourcer w/BIOS, UnPacker, ASMtool and View-It	299.75

Shipping & Handling: USA $5; Canada/Mexico $10; Other $15; CA Res. add sales tax; PS/2 trademark of IBM Corp.

NOT COPY PROTECTED **30-DAY MONEY-BACK GUARANTEE**

If within 30 days of purchase you find our product does not perform in accordance with our claims, call our customer service department and we will gladly arrange a refund.

VISA **1-800-662-8266** MasterCard

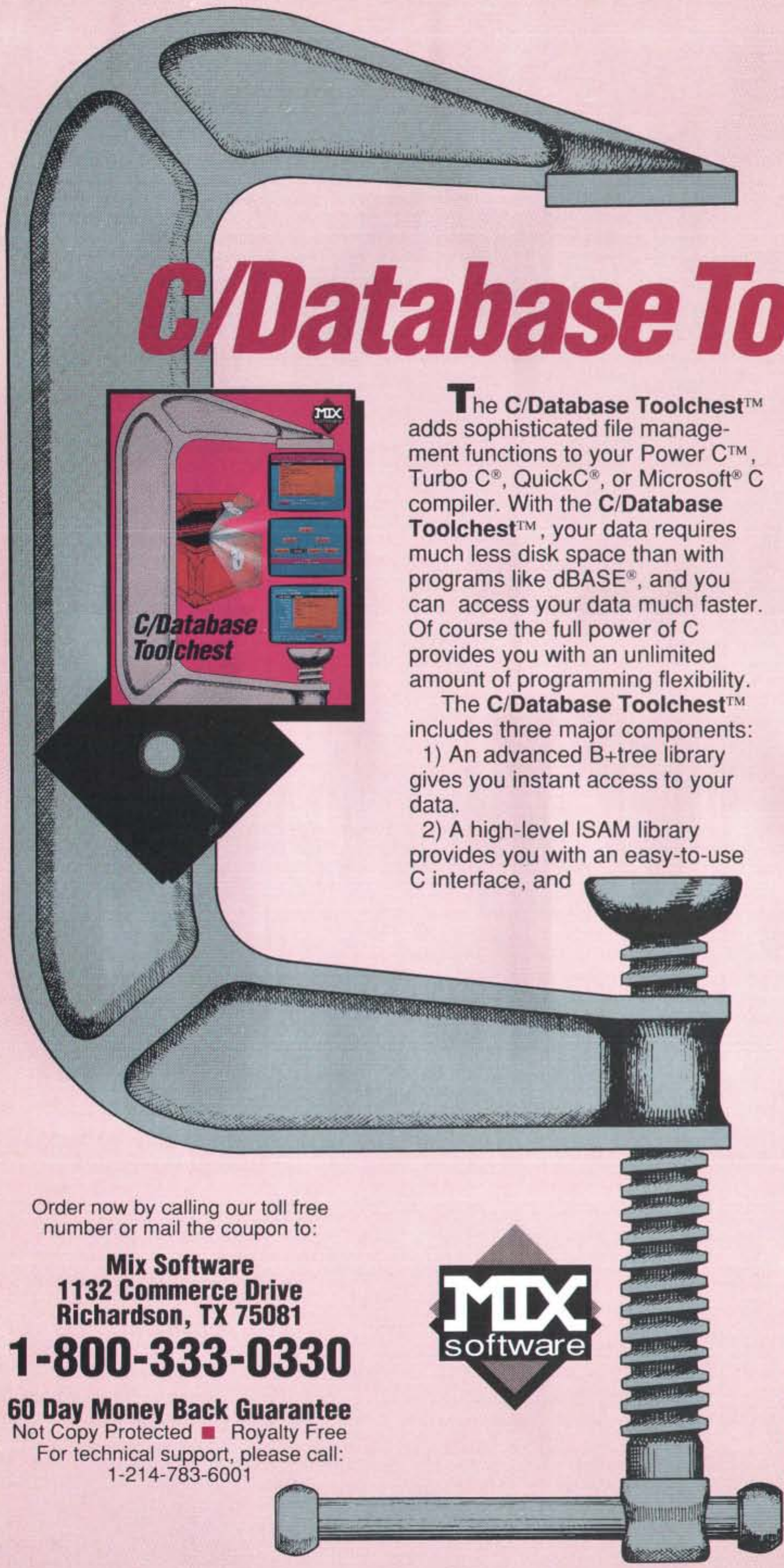
V COMMUNICATIONS, INC.

4320 Stevens Creek Blvd., Suite 275, Dept. DD6, San Jose, CA 95129 (408) 296-4224

CIRCLE NO. 245 ON READER SERVICE CARD

Turn your favorite C compiler into a powerful database manager with the

C/Database Toolchest



The **C/Database Toolchest**™ adds sophisticated file management functions to your Power C™, Turbo C®, QuickC®, or Microsoft® C compiler. With the **C/Database Toolchest**™, your data requires much less disk space than with programs like dBASE®, and you can access your data much faster. Of course the full power of C provides you with an unlimited amount of programming flexibility.

The **C/Database Toolchest**™ includes three major components:

1) An advanced B+tree library gives you instant access to your data.

2) A high-level ISAM library provides you with an easy-to-use C interface, and

3) A complete database manager (with C source code included) shows you how to create impressive applications.

You also receive a comprehensive 350 page manual and a utility for converting dBASE® files.

The **C/Database Toolchest**™ supports features that you'd expect to find only in products costing ten times as much. Advanced features include variable length records, variable length keys, multiple keys per index, and multiple indexes stored in a single file. Your data files can contain an unlimited number of records, and each record can be as large as 32K bytes in length.

About the only thing that the **C/Database Toolchest**™ doesn't do is cost you a lot of money. We've kept our price low so you can manage your budget as easily as your data.

Now Only $19.95!

Order Coupon

Name__________
Street__________
City__________
State__________ Zip__________
Telephone__________
Paying By____ Money Order____ Check
____ Visa ____ MC ____ AX ____ Disc.
Card#__________
Exp. Date__________
Disk Size______ 5¼" ______ 3½"

Qty.	Product	Price	Subtotal
____	C/Database Toolchest	$19.95	____
____	C/Database Library Source.. B+tree & ISAM library source code	$10.00	____
	Add Shipping ($5 USA, $20 Foreign)		____
	Texas Residents Add 8% Sales Tax		____
	Total Amount of Your Order		____

D

Order now by calling our toll free number or mail the coupon to:

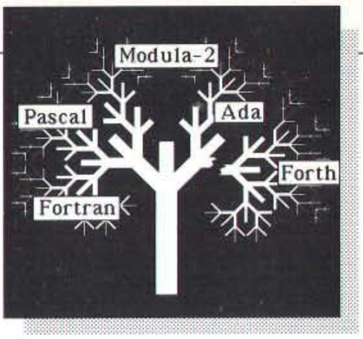
Mix Software
1132 Commerce Drive
Richardson, TX 75081

1-800-333-0330

60 Day Money Back Guarantee
Not Copy Protected ■ Royalty Free
For technical support, please call:
1-214-783-6001

MIX software

CIRCLE NO. 166 ON READER SERVICE CARD



Yes, Virginia, There Is a Xerox!

by Jeff Duntemann KI6RA

I damned near choked on my Cheerios the morning the *Mercury News* announced that Xerox was suing Apple over the Macintosh UI copyrights. Land o' Goshen! There is a Xerox after all! I had just about given up hope.

I have reason to feel specially about this whole situation. I spent ten years at Xerox (1974 – 1984) when much of their seminal research into user interface concepts took place. I was not directly involved with that research, but I spoke regularly with PARC people who were. I played with Smalltalk on the Alto prototype workstation, and I was one of the very first in Xerox MIS/DP trained on the Star workstation in early 1981. Unfortunately, I was one of the very few who ever saw the Star, much less worked on it. Had the Star caught on, Apple's suit against Microsoft would have been laughed out of court.

The Apple vs. Xerox Equation

For those of you who have been living in a Dewar's flask since 1985, let me recap: Last year Apple sued Microsoft and Hewlett Packard, claiming that Microsoft Windows 2.0 and HP New Wave violated Apple's visual copyrights on the Macintosh user interface. Earlier this year, parts of the suit concerning contractual technicalities were dismissed by the judge, leaving only the very large question of whether Windows 2.0 and New Wave indeed infringed on the Macintosh visual copyright.

Now Xerox, in its suit, claims that Apple appropriated the visual copyrights expressed in the Star workstation before the Macintosh was born, and that Xerox, as owner of said copyrights, is due royalties from the use of its interface.

There is a wonderful word to apply here, as ancient and ineluctable as fate itself: Checkmate. No matter which way Apple moves, it loses. Why? Because if Windows 2.0 infringes on the Macintosh, then the Macintosh infringes on the Star. I had to laugh at Apple's immediate protest that it was protecting its expression of the ideas pioneered by Xerox, and not the ideas themselves. Any three-legged salamander blind in one eye could tell you that Windows 2.0 and the Macintosh UI are not identical expressions, but instead are separate expressions of a common philosophy of user interaction. And if the similarities of expression between Windows 2.0 and the Mac are close enough to infringe on the Mac, then the Mac is similar enough to the Star to infringe upon the Star. Poof! Apple loses its claims of ownership.

It gets better. If Apple changes its mind and withdraws its Windows suit, Microsoft and HP have excellent grounds to complain that Apple brought the suit merely to stifle competition, leaving it open to prosecution under antitrust laws.

Couldn't happen to a nicer bunch of guys, eh?

Many people are saying that Xerox can't win its suit against Apple (or anyone else), because it waited far too long to bring the suit to begin with. This is true, but it may not matter. The scrutiny that Xerox will bring to bear on the whole question of who authored what and when will finally lay to rest the pernicious lie that Apple has some kind of ownership of anything with pull-down menus and overlapping windows.

Most people haven't seen the Star, so public opinion hasn't really gone up against Apple. I was there, however, when it happened, and I don't lie about these things: The Xerox Star is closer to the Mac than Windows is. That's the golden equation whose solution will, with luck, put windowing environments in the public domain and end this damned foolishness now and forever.

Thinking About Object Design

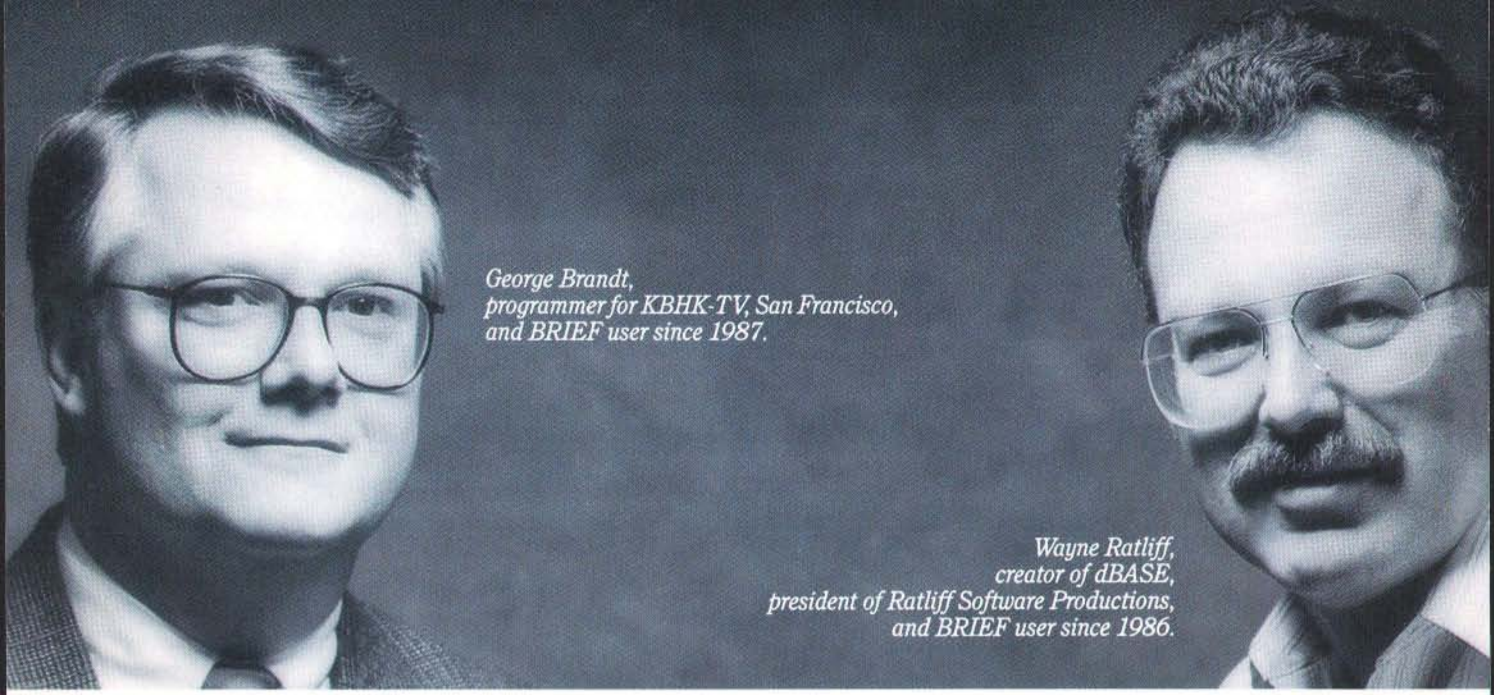
Just because you can swing a hammer doesn't mean you can design a house. That's the message that seems to be emerging from our first year as Object Pascal programmers. People are catching on to the syntactic nuances of object-oriented programming. Creating objects and methods is no longer quite the mystery it was in 1989. What does seem to be a mystery is the big picture of it all, containing sizeable questions like, "What should be an object? How do you divide your application into objects, and how do you apportion functionality to the objects in an object hierarchy?" This is the old architecture vs. carpentry duality made new again by the appearance of mass-market OOP languages. So put down that hammer for a bit, and let's talk about larger issues.

What Should be an Object?

Just after Turbo Pascal 5.5 and Quick Pascal appeared last May (almost simultaneously) people predictably began jumping on the OOP concept and taking it to absurd lengths, just for the exhilarating fun of it all. I recall hearing about guys who were creating objects out of ASCII characters, or even integers, because it seemed like the politically correct thing to do at the time.

As the vanguard soon discovered, making Pascal integers and characters into objects bought them almost nothing, and cost like hell. The first lesson we learned about OOP melted into conventional wisdom in record time: Atoms are atoms for a reason. Objects are a means of organizing complexity. Don't go looking for complexity where there isn't any. Molecules have to be made out of something.

My own rule of thumb comes down to this: Don't make objects out of anything that the compiler special-cases. This includes all fundamental types such as Boolean, Char, all numerics, and strings. In a language like Modula-2


George Brandt, programmer for KBHK-TV, San Francisco, and BRIEF user since 1987.
Wayne Ratliff, creator of dBASE, president of Ratliff Software Productions, and BRIEF user since 1986.

Of course they need completely different kinds of editors. That's why they both use BRIEF.

Let's face it. There's no such thing as two programmers who think alike. Or who work alike.

Because everybody's got their own style, their own habits. Their own quirky little requirements. As well they should. So, where are you going to find one editor that satisfies everyone? Simple. Look inside a BRIEF® box.

It's the right editor, right out of the box.

What you'll find inside is an enormously powerful, astonishingly easy, and highly-evolved program editor. One that works at blinding speed. And that's loaded with features designed to help you be more productive. One guaranteed to give you excellent results within one short hour of booting up the first time.

You'll find features like automatic indenting, infinite windowing, and powerful regular expression searching that goes well beyond the ordinary text-matching you're probably used to.

You'll also find the features that made BRIEF famous. Including our near-miraculous UNDO, that doesn't just un-delete your last keystroke. It actually reverses the effect of any command. So you can explore, experiment, even make gigantic mistakes with hideous consequences, and just "undo" them. And then continue to undo, backing-up through 300 steps.

What you want is what you get.

What you'll also find is an editor that delights in adapting itself to you. With the simple, menu-driven SETUP program, you can quickly customize indenting style, support for any of 36 compilers, keyboard configuration, rate of keyboard auto-repeat, and much more.

You can even edit BRIEF's macros to suit your taste. Or write your own in a snap.

Computer Language said that BRIEF is capable of "almost complete customization." And with new release 3.0, BRIEF is even more powerful and flexible.

For every programmer, it's a different BRIEF.

At last count, there were over 52,000 copies of BRIEF in the field, making life easier and working time more productive for all kinds of programmers—working in all kinds of languages. Programmers as different as Wayne Ratliff is from George Brandt. Which means there are over 52,000 different ways of working, and BRIEF fits them all.

Why not get BRIEF going to work for you? There's no risk. In fact, if you're not completely satisfied that BRIEF is exactly the editor you've been looking for, just return it within 30 days, and we'll return your money. No questions asked.

New BRIEF 3.0 highlights

- With our new C-like macro language, CBRIEF, C programmers can write macros on the fly and not have to switch between C and a different macro language.
- The new source level debugger works with both the original BRIEF syntax and the new CBRIEF macro language.
- Maximum lines per file has been increased to 4 billion.
- We've added to and improved template editing and smart indenting for ADA, Cobol, BASIC, FORTRAN, Modula-2, Pascal, and C.
- We've added editable, multiple keystroke macros that can be saved and restored.
- You can even save and restore your window set-ups.

Plus the features that made BRIEF famous.

- 300-level UNDO safety net
- Flexible windowing, unlimited quantity
- Unlimited number of files
- File size limited only by disk space
- Regular expression search and replace
- BRIEF's LISP-like macro language
- Toll-free tech support

BRIEF 3.0
The Programmer's Editor

Solution Systems™ 541 Main St., Suite 410 South Weymouth, MA 02190 FAX 617-337-7719

Try BRIEF at our risk. $199 plus $6 shipping (U.S. and Canada). 30 day money back guarantee.

Name ______
Company ______
Address ______
City ______ State ______ Zip ______
Phone ______
Paying by ☐ Check ☐ MC ☐ VISA
Card # ______ Exp. date ______
Signature ______
Disk size ☐ 5-1/4" ☐ 3-1/2" Format ☐ DOS ☐ OS/2

Mail this coupon or call toll-free today.

800-821-2492
Extension 442

Massachusetts & outside U.S. call 617-337-6963

International Distributors: England (0763) 73457 West Germany (069) 34 72 26 France 47.70.47.70 Australia (03) 663 6580 Denmark (02) 88 72 49 Norway 03-89 22 40 Austria 1 31 70 36 Italy (02) 837.83.41 Holland (020) 668 5606 Israel 4 238757

CIRCLE NO. 229 ON READER SERVICE CARD

where strings have no special status to the compiler, a string object might make sense. Certainly a long string type (in which you can keep more than 255 characters) is a prime candidate for objecthood. Strings, however, have special privileges with respect to text files, *Readln* and *Writeln*, and a whole slew of built-in string-handling routines. Furthermore, strings may be returned as function results, a capability I would be extremely reluctant to give up. Encasing a Pascal string within an object wrapper actually adds complexity to an application, and that's not how the game is supposed to be played.

An Exercise in Modeling Reality

When I wrote the tutorials in the Turbo Pascal 5.5 *OOP Guide*, this sort of question hadn't occurred to me, as I was nearly as new to Object Pascal as the rest of the PC world and was running but a step and a half ahead of the Comprehension Wolves through the whole project. What I knew of objects came to me from Smalltalk, in which everything is an object, not excluding atomic particles such as characters and integers. But my Smalltalk experience kept me in good stead when I had to confront questions from readers like, "How should I divide an application up into objects?" and, "How do I define the boundaries between objects?" There's no easy answer this time, but I've worked with a principle that I consider very effective: Whenever possible, reflect in your objects the divisions and relationships seen in reality. In other words, make your OOP programs simulations even when they're not really simulations.

There I go again, getting Eastern on you. Time to Get Real.

Artifacts, Not Actions

Objects were originally conceived by Simula's architects as models of elements of reality. Reality is not soup. Reality is lumpy, and the lumps, seen apart, are identifiable things with identifiable functions. A lamp is a thing that emits light when you hit its buttons correctly. A carpet keeps the oak floors from getting scraped. Your Peter Norton mug is a mechanism that cooperates with gravity to keep your coffee out of your lap. Every lump in reality has a name and a job, though the necessity of some, like city planners and the rock band Metallica, is seriously open to question.

All of this is second nature to anyone who has survived his or her Terrible Twos, but I remind you of the nature of reality because I want you to apply it to the way you arrange the concept of a program in your mind. Too many programmers make their programs like you'd make soup, by tossing in a pinch of this and half a cup of that and stirring it around until it becomes relatively uniform in color and texture and ceases to be lethal.

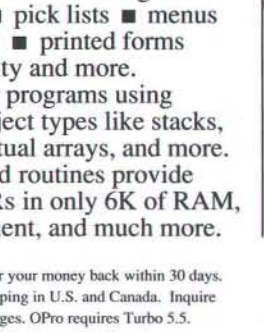
Because programming has always been an action-oriented process, programmers often begin by asking what steps constitute the larger action of the program. If you have gotten to this point, the cooking has already begun, and you're halfway to soup already. Never forget this: A program is not an action. A program is an artifact. Once you throw the artifact into your conceptual duck-press and squeeze it down to a statement of action, you've lost a great deal of valuable information about the idea of the program. To recap in a slightly different way: A statement of what a program does is far less useful than a statement of what the program is. The best example has been staring you in the face for years. Tell me first what a spreadsheet program does, and then when you've given up on that,

Save "Man-Years of Effort" with Turbo 5.5

Don't Start from Scratch with Object-Oriented Pascal

Object Professional is a huge library of over 200 object types and 2000 methods that will multiply your productivity. Window object types let you use overlapping and resizeable windows. The windows include ■ scrolling data entry screens ■ pick lists ■ menus ■ file selection ■ printed forms ■ help capability and more.

Build your programs using proven data object types like stacks, linked lists, virtual arrays, and more. System-oriented routines provide swappable TSRs in only 6K of RAM, EMS management, and much more.



Object Professional includes clear, comprehensive documentation, on-line help, full source code, technical support, and hot demo programs. Pay NO royalties. You'll get up to speed fast with OOP!

"The range of objects is fantastic. Object Professional could literally save you man-years of effort."

Jeff Duntemann

Object Professional 1.0, only $150.

A Multi-User B-Tree Toolkit

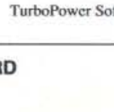
Write powerful network compatible databases faster and easier using **B-Tree Filer 5.0**. You'll have the fastest, safest, most flexible databases – no rigid structure, no TSR hassles, no running out of files. And they're compatible with Novell, 3Com, MS-NET, and others.

You get ■ Fixed and variable length records ■ Two billion records per database ■ Up to 100 indexes per index file ■ Fail-safe mode with journaling ■ Units for sorting, browsing, reindexing, and network control.

B-Tree Filer includes full source code, documentation, technical support, and you pay NO royalties.

"B-Tree Filer... a well rounded, feature-rich approach to B-Tree databases."

Computer Language, 1/90



B-Tree Filer 5.0, only $125. (single user)
With network support, $175.

Call toll-free to order. 1-800-333-4160

Satisfaction guaranteed or your money back within 30 days. Add $5 per order for shipping in U.S. and Canada. Inquire about other shipping charges. OPro requires Turbo 5.5. BTF requires Turbo 4.0, 5.0, 5.5, or QuickPascal.

TURBO Power

8AM - 5PM PST Monday through Friday, USA & Canada. For more information call (408) 438-8608. Fax: (408) 438-8610. TurboPower Software PO Box 66747 Scotts Valley, CA 95067-0747

CIRCLE NO. 204 ON READER SERVICE CARD

tell me what a spreadsheet is. A spreadsheet does a whole host of things, but what a spreadsheet is is a model of the classic accountant's paper ledger sheet. By now, the lights should be coming on.

Collecting Stamps

Seeing artifacts where you used to see actions is perhaps the most important skill to be learned in picking up OOP. Let's practice a little by identifying a program artifact and giving it reality as an object.

The artifact in question is the humble moment, the point in time at which something happens. In certain kinds of programming, particularly business or financial programming, the time and date when a transaction happens can be almost as important as the transaction itself.

You've seen evidence of a moment in every DOS directory listing: The date and time when the file was last modified. Every directory entry on a DOS disk has a time stamp and a date stamp, which are sometimes lumped together and simply called the time stamp.

That's your artifact. To avoid confusion between its time and date components, (because it incorporates both) I call it a "when stamp." Any time you need to retain an expression of the moment that something happened in your program, you can create a when stamp object for it.

Objects embody both data (the object's "state") and the actions necessary to manipulate that data. The data in a when stamp is straightforward: Some single, unambiguous expression of both clock time and calendar date.

DOS has such an expression in its directory time and date stamps. It makes sense to use what DOS uses, (especially when DOS can be made to do some of our work for us) so let's agree to keep our when stamp as DOS-compatible as possible. The DOS time stamp and date stamp are both 16-bit words, so we can combine them into a single 32-bit type in Object Pascal. The obvious candidate is *LongInt*, the 32-bit long integer type.

Having the when stamp stored in a single numeric type carries the side benefit that two when stamps can be compared for time priority just by comparing with the numeric greater than and less than operators.

Time values are never negative, so you might begin to worry about the sign bit in a long integer, which is a signed type. As it happens, you will have to worry, but not for a very long time. (More on this later — nonetheless, I still wish that Pascal had a *LongWord* type.)

Once we get the expression of time and date, we'll need methods to manipulate it. But before you start thinking about methods, get your data and its representation in order.

Encoding Time and Date, DOS-style

Understanding how our When object works depends heavily on knowing how DOS encodes its time and date stamps in their 16-bit words.

If we used neat decimal star dates like Captain Kirk, we'd be better off, but alas, a date is a set of three separate numeric quantities: Year, an ascending value that never repeats, month, a value that repeats regularly through a cycle of 12, and day, a value that repeats through a cycle of 28, 29, 30, or 31. This makes date arithmetic ugly unless the date is encoded as a single numeric value. The means is this: Express year, month, and date as binary quantities, then line them up in a single 16-bit word such that the year portion occupies the highest-order bits, the month portion occupies the next-highest order bits, and the day portion occupies the lowest-order bits. Done this way, two date stamps with different years will always compare correctly regardless of month or date; two stamps with identical years will always compare correctly on the basis of month, regardless of date, and so on. Which bits relate to year, month, and date is shown in Figure 1.

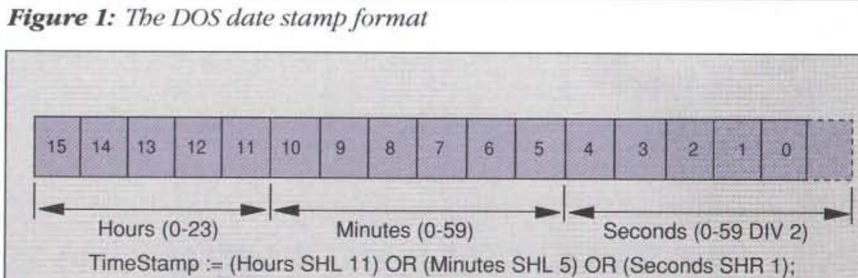
Note that the year is encoded in a peculiar (and I think unfortunate) way: As an offset from the year 1980. 1980 is thus year 0, 1981 year 1, and so on. This allows the year to be encoded in only a few bits, leaving plenty of room for the month and day in a 16-bit word. The downside is that you can't encode your birthday as a time stamp, since you were probably born considerably prior to 1980. (I was, and have the hairline to prove it.) Seriously, this limits the use of when stamps to things that occur in true calendar time while the computer is in use, and not to encode events that happened long ago. I was a little concerned about the use of a signed type to hold the when stamp. At some point, the bit encoding of the time and date will set the high bit in the long integer, turning the value negative in the eyes of the run-time library. This blows any possibility of valid comparisons out the window, because a later value that is negative will be seen as less (and hence earlier) than any positive stamp. However, when I did the math I found that the stamp does not turn negative until December 31, 2043, by which time I will either be dead or perfecting zero-G lovemaking techniques out past the orbit of Mars.

17 Pounds of Kitty Litter in a 16-pound Bag

The DOS time stamp presents a problem. No matter how you encode the hours, minutes, and seconds (and forget hundredths here) you end up with a minimum of 17 bits: Six for minutes (0 – 59); six for seconds (0 – 59), and five for hours (0 – 23).

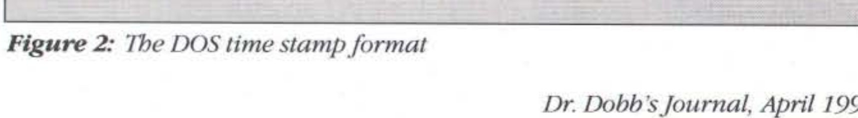
You can usually cram 17 pounds of kitty litter into a 16-pound bag by shaking things around a little. Not so in the bit game — we pack 'em tight. Something has to give a little, so what we do is ignore every other second. This cuts the number of bits required to encode seconds to five, and 16 bits will suddenly hold the stamp. See Figure 2.

Note the way the stamp is encoded



```
DateStamp := (Year SHL 9) OR (Month SHL 5) OR Day;
```

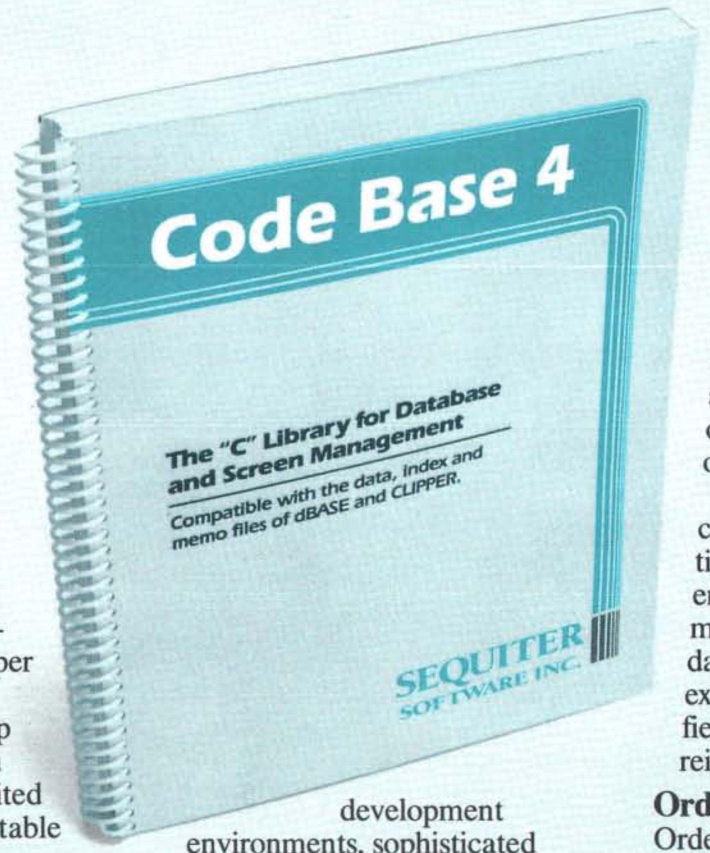
Figure 1: *The DOS date stamp format*



```
TimeStamp := (Hours SHL 11) OR (Minutes SHL 5) OR (Seconds SHR 1);
```

Figure 2: *The DOS time stamp format*

C Why dBASE programmers are excited!


Code Base 4
The "C" Library for Database and Screen Management
Compatible with the data, index and memo files of dBASE and CLIPPER.
SEQUITER SOFTWARE INC.

Build a multi-user, 85K, dBASE compatible application using pulldown menus, popup windows, and data entry from pick lists.

Portable
When you are done, port your application to Unix, Microsoft Windows and OS/2 without modifying a single line of code.

Then watch as your application runs many times faster than corresponding dBASE, Clipper or Foxbase programs.

Finally, you can keep all the profits after you have distributed unlimited numbers of your executable programs royalty free.

Compatible
Code Base 4 lets you access and modify the data, index and memo files of dBASE III, dBASE IV, or Clipper. Consquently, you can take advantage of dBASE compatible tools such as R&R Relational Report Writer.

Switch between Turbo C, Quick C, and Microsoft C. Take advantage of integrated development environments, sophisticated debuggers, and programs which compile and link in seconds.

Learn Code Base 4 by consulting the comprehensive 206 page user's guide while interactively executing Code Base 4 routines from a learning utility. Then try example programs from the diskettes or the user's guide. You will easily remember the Code Base 4 routines which correspond directly to familiar dBASE commands.

Source Included
As you become an expert Code Base 4 user, you will find yourself examining the source code as you read about the internal operating principles of Code Base 4.

Enjoy the benefits of complete dBASE functionality, including data entry, windows, menus, multiple index files per database, dBASE expression evaluation, fields, filters, relations, reindexing, and editing.

Order Today
Order Code Base 4 at $295 and you will soon know why Sequiter Software Inc. and most software dealers are happy to give a 30 day money back guarantee!!

Call (403) 439-8171
Fax (403) 433-7460

SEQUITER
SOFTWARE INC.
P.O. Box 5659, Station L
Edmonton, Alberta
Canada T6C 4G1

KEDIT 4.0
NEW TEXT EDITING POWER

KEDIT is a powerful, general purpose text editor for the IBM PC, PS/2, and compatibles that supports the editing of multiple ASCII files in multiple windows, sophisticated block move and copy operations, and reprogrammability of the keyboard through keyboard macros. The latest version, KEDIT 4.0, provides these new features:

- **EMS expanded memory support** that allows editing of larger files.
- **Selective line editing capability** compatible with XEDIT's 'ALL' command.
- **Enhanced programmability** with improved interfaces to Mansfield's Personal REXX macro processor and a built-in REXX subset (KEXX).
- **IBM VGA and Enhanced Keyboard support.**

Proven in operation since 1983, KEDIT is unique in its support of many of the commands and features found in XEDIT, IBM's mainframe text editor for the VM/CMS system. Versions of KEDIT 4.0 are available for both DOS and OS/2.

MANSFIELD
Software Group
P.O. Box 532, Storrs CT 06268
(203) 429-8402

KEDIT Version 4.0 is available at $150; OS/2 version (includes DOS version) is $175. Add $3 shipping. MC, VISA, American Express.

KEDIT is a trademark of the Mansfield Software Group, Inc.; IBM and IBM PC and PS/2 are trademarks of International Business Machines.

CIRCLE NO. 289 ON READER SERVICE CARD

PROLIFIC++
INTEK C++ 2.0

- ***Compile larger programs on 80286 or 80386.*** Translator runs in 80286 or 80386 protected mode; accesses all of memory from DOS.
- ***Choose the compiler you prefer.*** Use INTEK C++ with *any* standard (K&R) C compiler. Library support is included for the six most popular DOS compilers. 80386 version supports 80386 native C compilers from MetaWare and Watcom.
- ***Windows? No problem!*** Extended keyword compatibility allows you to compile for Microsoft Windows®. OS/2® native version coming soon!
- ***Simplify your development environment.*** Use INTEK C++ to cross-compile from Unix® System V to DOS, so you can keep everything in one place.

80286 DOS	$249
80386 DOS	$495
80386 Unix	$495
OS/2	call

Authorized Dealer:
The Programmer's Shop
1-800-421-8006

INTEK
INTEGRATION TECHNOLOGIES
1400-112th Avenue SE, Suite 202
Bellevue, Washington 98004
(206) 455-9935 FAX: (206) 455-9934
Usenet: uunet!intek01!info

INTEK C++ is derived from AT&T version 2.0 and is licensed from AT&T. Windows is a trademark of Microsoft. OS/2 is a registered trademark of IBM. Unix is a registered trademark of AT&T. MetaWare, Microsoft, and Watcom are trademarks of their respective companies.

CIRCLE NO. 119 ON READER SERVICE CARD

(continued from page 138)
in the figure: Whereas the hours and minutes values are shifted left to move them toward the high end of the word, the seconds value is shifted right by one bit, which bumps that bit off into Peoria and out of our hair. Truncating the seconds by half means that two events occurring less than two seconds apart will probably resolve to identical time stamps, depending on their synchronization with the system clock. You have to keep this in mind and avoid designing time stamps encoded this way into applications where stampable events happen in quick succession. Note also that your seconds value will always be an even number, because the 1 bit that can make it odd is not stored in the stamp.

To Represent or Calculate?

At the heart of it, then, a *When* object consists of a long integer time/date stamp encoded as explained earlier. If you only needed to compare two stamps for time order, this would be enough. However, you may need to access the seconds value separately, or the hours or months, and you may want to display time or date or both in some generally understood ASCII form. There are two ways to do this:

- Add separate fields to the object to contain distinct hours, minutes, and seconds; and years, months, and days. Then recalculate and update those fields any time the stamp itself changes.
- Leave the time/date stamp as the sole data field in the object, and calculate any component value whenever a user of the object requests it.

The first option buys speed at the expense of space, and the second buys space at the expense of speed. Which do you use? Well, do you need more speed or space?

I'm not just being glib here. There's no single answer. You as the object's designer have to keep track of your own particular needs. I tend to write small programs, and I like fast ones, so my own first impulse is to throw memory at problems to make them faster. This is the design choice I made in implementing the *When* object, as shown in Listing One, page 150.

The *When* object has as its central data item the *WhenStamp* long integer field. It also has separate fields for year, month, day, day of week, hours, minutes, and seconds. I added an ASCII representation of time identical to that used in the DOS DIR command, as well as two ASCII date representations: One in the form *yy/mm/dd*, and the other in the form "Thursday, June 29, 1989." This adds 94 bytes to the size of the object, but I did it with eyes open and decided that the benefits were worth the cost. There was one exception. I chose not to provide a separate 16-bit time and date stamp, as I had originally considered doing, because returning one half or the other of a long integer can be done without any significant calculation overhead, just by typecasting. Notice the definition of *WhenUnion*, private to the unit:

```
WhenUnion =
  RECORD
    TimePart : Word;
    DatePart : Word;
  END;
```

WhenUnion is the same size as *LongInt*, so you can use a value typecast to access either the time or date portion of the combined time/date stamp:

```
TimeStamp :=
  WhenUnion(WhenStamp).TimePart;
```

It's that easy, and saves you 4 bytes of memory at the cost of no calculation at all. I like that sort of deal — kind of like insider trading in Silicon Valley.

Choosing Your Methods

Once you've decided what your object is, you can begin working out what it does; that is, design its suite of methods. I wanted my when stamp object to be easily updated, so I provided numerous methods for changing the value of the *WhenStamp* field. All the methods beginning with *Put* change the value of *WhenStamp*, and all the methods beginning with *Get* return some value from the object. This is a good naming convention when designing objects, and I recommend it. The *PutNow* method is simple but extremely useful: It reads the current value of the PC's clock/calendar and applies the current time and date to the when stamp. The other *Put* methods alter the value of the when stamp by providing a new value for either the whole stamp (*PutWhenStamp*) or some component value of the stamp, such as year, month, or hours. How many such methods you build into an object depends on how you intend to use the object. My recommendation is to build more into the object than you may need, and reexamine the design some time down the road. You can always strip out what you haven't used . . . but you never know when some flash of insight will allow you to find a use for a method that you hadn't imagined when you originally wrote it!

Windows and OS/2 PM Programmers

We want you to get this message

Eikon's Control Paks™ are collections of graphical, high-level, reusable software objects designed for use by developers of Microsoft Windows™ and OS/2™ Presentation Manager™ software. Each royalty-free Control Pak contains three control classes, all supporting different visual styles. The Standard Pak includes Arrow, Palette, and Picture controls. The Tools Pak includes Ruler, Slider, and Toolbox controls.

Suggested U.S. List Prices	*Without Code*	*With C Code*
Standard Pak *(Windows)*	$125	$475
Standard Pak *(OS/2 PM)*	$225	$675
Tools Pak *(Windows)*	$175	$525

INTER-OFFICE MEMO

To: Marion
From: Steve
Re: R & D Schedule

GREAT JOB!

Eikon's control paks are superb! They've saved us a great deal of development time -- and the cost was more than worth it!

Thanks for getting your job done ahead of schedule!

P.S. Have you checked out any of Eikon's other R&D tools?

We've written the code ~ so you don't have to!

ORDER TODAY!
Call (415) 349-4664

Eikon Systems, Inc.
989 E. Hillsdale Blvd, Suite 260
Foster City, California, USA 94404

EIKON

CIRCLE NO. 417 ON READER SERVICE CARD

The
Heap Expander™
version 2.05

Now your programs can have virtually unlimited heap space using expanded memory, extended memory and disk space. The Heap Expander's initialization code checks the system's resources and uses whatever is available.

still $79.95*

- Uses LIM-standard expanded memory if present.
- Uses AT-style extended memory if present.
- Swaps data to disk as needed.

Libraries and Source Code for:

- Turbo C
- Microsoft C 4.0 , 5.0 and 5.1

Requires an IBM PC, XT, AT, or close compatible with MS-DOS or PC-DOS version 2.0 or above

MC/VISA/COD call
1-800-248-1045 x 100 (US)
1-800-952-5560 x 100 (Idaho)

Foreign customers add $6.00 for shipping and handling.

The Tool Makers

P.O. Box 2151
Santa Cruz, CA 95063
408-458-0690

CIRCLE NO. 239 ON READER SERVICE CARD

The Encapsulation Question

The "pure" object-oriented languages such as Smalltalk and Actor both allow and enforce total encapsulation of an object's data. You cannot directly reference an object's data from outside the object or its descendants. Object Pascal allows total encapsulation, in that you can voluntarily provide a separate method to return the value of each field in the object. Nothing enforces this, however — you can reference any field in any object from anywhere in your program. Purists will say, "So what?" Give 'em the methods anyway, and say hands off the data. Such hardnosedness gives you the considerable benefit of being able to change the actual representation of the data within an object without breaking the code that uses the object. For example, if I forbade direct references to the data inside *When*, I could come up with my own time stamp format that was truly universal and did not ignore the existence of years prior to 1980 as well as every other tick of the clock.

If you needed to provide stamps for old financial records, birthdays, and such (as in a life insurance application) you'd be far better off going this route. My own need for time stamps, however, was limited to stamping transactions occurring at the current moment, so I made the decision not to add the additional level of complication to When. I also wanted to retain full compatibility with the DOS directory time and date stamps, and the easiest way to do that is simply to represent the time and date the same way DOS does. Given these assumptions, I created When to allow direct read access to all the data fields.

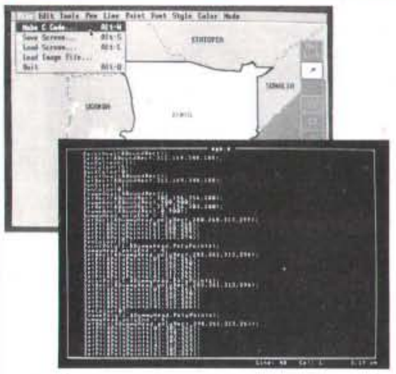
Note that I said read. Writing directly to the fields is not a good idea, because all the fields but *WhenStamp* are actually component values of *WhenStamp*, and if you were to change the *Month* field without changing *WhenStamp*, the two would be "out of sync." The Put methods were designed so that changing any part of the time stamp recalculates all component values relating to that part of the time stamp. For example, changing the date half of the when stamp recalculates the *Year*, *Month*, *Day*, *DayOfWeek*, *DateString*, and *LongDateString* fields, without affecting *Hours*, *Minutes*, *Seconds*, or *TimeString*.

You need to be aware of such dependencies when deciding how to allow updating of an object's data. Calculating component values from a master field such as *WhenStamp* whenever they are needed avoids problems like this, if you can tolerate the loss of performance.

Private Parts

For reasons unknown, many newcomers to OOP get the notion in their noggins that all of an object's processing must be confined to its methods, and that while methods can call one another, methods somehow cannot call other procedures and functions unrelated to the object. Not true! A method can call any routine within its scope, just as any nonmethod procedure or function can. Furthermore, there is no way to declare a "private" method in Object Pascal. A private method would be one that could be called by other methods within the object, but that could not be accessed from outside the object. If there is any private processing to be done by an object, you can do what I've done with When and place the object definition in a unit — with private procedures and functions fully declared and defined within the implementation section. Such procedures and functions may be called by the object's methods but are not available to anyone outside the unit itself. Similarly, the *MonthTags* and *DayTags* constants exist for the convenience of the object and are not needed by users of

Drawbridge™



Reduce graphics development time with Drawbridge–the source code generating graphics editor.

★ Increases productivity
★ Object-oriented editing
★ Source code generation
★ Uses library features
★ Available for C or Pascal

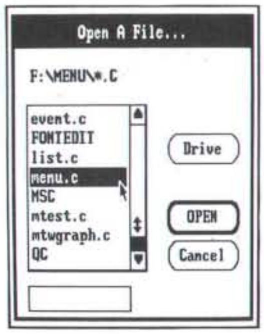
Drawbridge for:

MetaWINDOW	$189
Borland BGI	$89
Microsoft graphics library	$89

Chatterbox™

Create a graphical user interface quickly and easily with the Chatterbox interface library.

★ Works in graphics mode
★ Event Queue Management
★ Pull-Down Menu System
★ Dialog Box Manipulation
★ Interfaces to most graphics libraries



Chatterbox Library $189

Order today! Call: (217) 359-1878

Courseware Applications, Inc.

475 Devonshire Drive • Champaign, IL 61820

CIRCLE NO. 110 ON READER SERVICE CARD

the object, so I've declared them privately, within the implementation section. Ditto with *WhenUnion*. As long as you provide functions to return separate 16-bit values for the time stamp and date stamp portions of the when stamp (as I did with *GetTimeStamp* and *GetDateStamp*), there's no need to give the user the *WhenUnion* definition. Half the battle of programming is masking complexity, so keep an object's private parts under a bushel basket where they won't be stepped on or misused by the ignorant and the unwary.

One of the private routines is a day-of-the-week calculator using an algorithm called "Zeller's Congruence." I've had this routine in my files for so long I no longer remember who coded it up and gave it to me, and I categorically do not understand how it works . . . but it does work.

Homework Assignments

That's how I went about designing a very simple object. To make sure it sinks in, do the following things for practice:

• Recode the *When* object such that the only data field is the long integer *WhenStamp*, with all other component values and alternate representations calculated as they are needed, by new *Get* methods. This shouldn't involve much new code at all, but will involve moving existing code around a lot. While you're at it, you might add a simple function to return the full long integer value of *WhenStamp*, giving you total encapsulation. You might like this more compact version of When more than mine. So be it. I'm not you. (And I suspect we both should be glad. . . .)

• Write new methods to add or subtract specific values from the when stamp. You might want to create a when stamp containing a value exactly 30 days from right now, and then monitor that stamp over the coming month to ensure that something necessary happens at that moment. This requires a way of adding 30 days to the long integer *WhenStamp* value. It's fairly easy . . . do it!

Object Design Recap

To summarize:

An object is an artifact, not an action. Look at the way the universe is broken down into components, and learn to think of your programs as built of component parts in much that same way.

Once you've identified such a software artifact, decide how to represent its data first. Only once you have the data design down should you begin to ponder what methods it needs to serve that data. Remember, Data is Boss in object land.

It's possible and often desirable to disallow any direct access to an object's data. This frees you to change the way the data is stored within the object without breaking code that uses the object. Remember, however, that this can add significant performance handicaps to code that uses your objects. Keep the tradeoffs in mind, but (as the cricket kept saying) always let your conscience be your guide.

Also remember that this is just a first lesson. We haven't even begun to address the issues presented by inheritance or, lord knows, polymorphism. All in good time.

Have You Seen this Book?

Maybe you all can help me out a little. My newest book, *Assembly Language From Square One*, (Scott, Foresman & Company) has been off the presses for some time, and I have yet to see it in any store. If you have seen it anywhere, drop me a postcard at *DDJ* and tell me what store is carrying it. The computer book distribution system seems to be breaking down in recent months, much as it did in 1984. Huge numbers of titles, most fit only to hang in a Tennessee outhouse, have been pouring into the retail channel lately, and it's gotten me (and some other well-known authors) more than a little worried. We may not be able to change the system, but we'd at very least like to know what books are going where.

Thanks.

Write to Jeff Duntemann on MCI Mail as JDuntemann, or on CompuServe to ID 76117,1426.

Availability

All source code is available on a single disk and online. To order the disk, send $14.95 (Calif. residents add sales tax) to *Dr. Dobb's Journal*, 501 Galveston Dr., Redwood City, CA 94063, or call 800-356-2002 (from inside Calif.) or 800-533-4372 (from outside Calif.). Please specify the issue number and format (MS-DOS, Macintosh, Kaypro). Source code is also available online through the *DDJ* Forum on CompuServe (type GO DDJ). The *DDJ* Listing Service (603-882-1599) supports 300/1200/2400 baud, 8-data bits, no parity, 1-stop bit. Press SPACEBAR when the system answers, type: listings (lowercase) at the log-in prompt.

DDJ

(Listings begin on page 150.)

Vote for your favorite feature/article.
Circle Reader Service **No. 11.**

Multitasking

for the IBM-PC/XT/AT
with

Multidos Plus

Ideal for developing process control, data acquisition, communications, and other applications which require a multitasking environment. Check these features which make Multidos Plus an unbeatable value:

- Run up to 32 programs concurrently
- Your software continues to run under PC-DOS or MS-DOS. No need to learn a new operating system.
- Use the compilers you already have. Supports software written in most compiled languages or assembly language.
- Operator commands to load and run programs, change priority, check program status, and to abort/suspend/resume task execution.
- Optional independent foreground and background displays.
- Programmatic interface through INT 15H for the following:
  - * Intertask messaging to send, receive, check or wait on 63 different message queues.
  - * Task control by means of semaphores. Get, release, check, or wait on semaphores.
  - * Change task priority.
  - * Suspend task for specified interval.
  - * Spawn and terminate external and internal tasks.
  - * Disable/enable multitasking.

Requires: IBM PC/XT/AT or true clone. MS-DOS (or PC-DOS) 2.0 or later. Enough memory to hold Multidos Plus (64KB) and all your application programs. You may also need 4 or 16KB of memory for "hidden screens" for each active task.

$24.95 or

$99.95 with source code (written in Microsoft QuickC 2.00 and Assembler)

To order call (800) 678-2141. FAX (508) 655-8860. For technical information and foreign orders call (508) 651-0091. Visa and Mastercard accepted. Or send check or money order (Drawn on U.S. banks only) to:

Nanosoft

13 Westfield Road
Natick, MA 01760

Add $2.00 for shipping & handling.
Foreign orders add $5.00 for shipping & handling.
COD orders add additional $3.00
Massachusetts shipments add 5% sales tax.

CIRCLE NO. 347 ON READER SERVICE CARD

Listing One *(Text begins on page 127.)*

```
/* --------------------- csort.h -------------------------- */

#define NOFLDS 5         /* maximum number of fields to sort   */
#define MOSTMEM 50000U   /* most memory for sort buffer        */
#define LEASTMEM 10240   /* least memory for sort buffer       */

struct s_prm {                 /* sort parameters              */
    int rc_len;                /* record length                */
    struct  {
        int f_pos;             /* 1st position of field (rel 1)  */
        int f_len;             /* length of field                */
        char ad;               /* a = ascending; d = descending  */
    } s_fld [NOFLDS];          /* one per field                  */
};

struct bp   {            /* one for each sequence in merge buffer */
    char *rc;            /* -> record in merge buffer             */
    int rbuf;            /* records left in buffer this sequence  */
    int rdsk;            /* records left on disk this sequence    */
};

int init_sort(struct s_prm *prms);  /* Initialize the sort     */
void sort(char *rcd);               /* Pass records to Sort    */
char *sort_op(void);                /* Retrieve sorted records*/
void sort_stats(void);              /* Display sort statistics*/
```

End Listing One

Listing Two

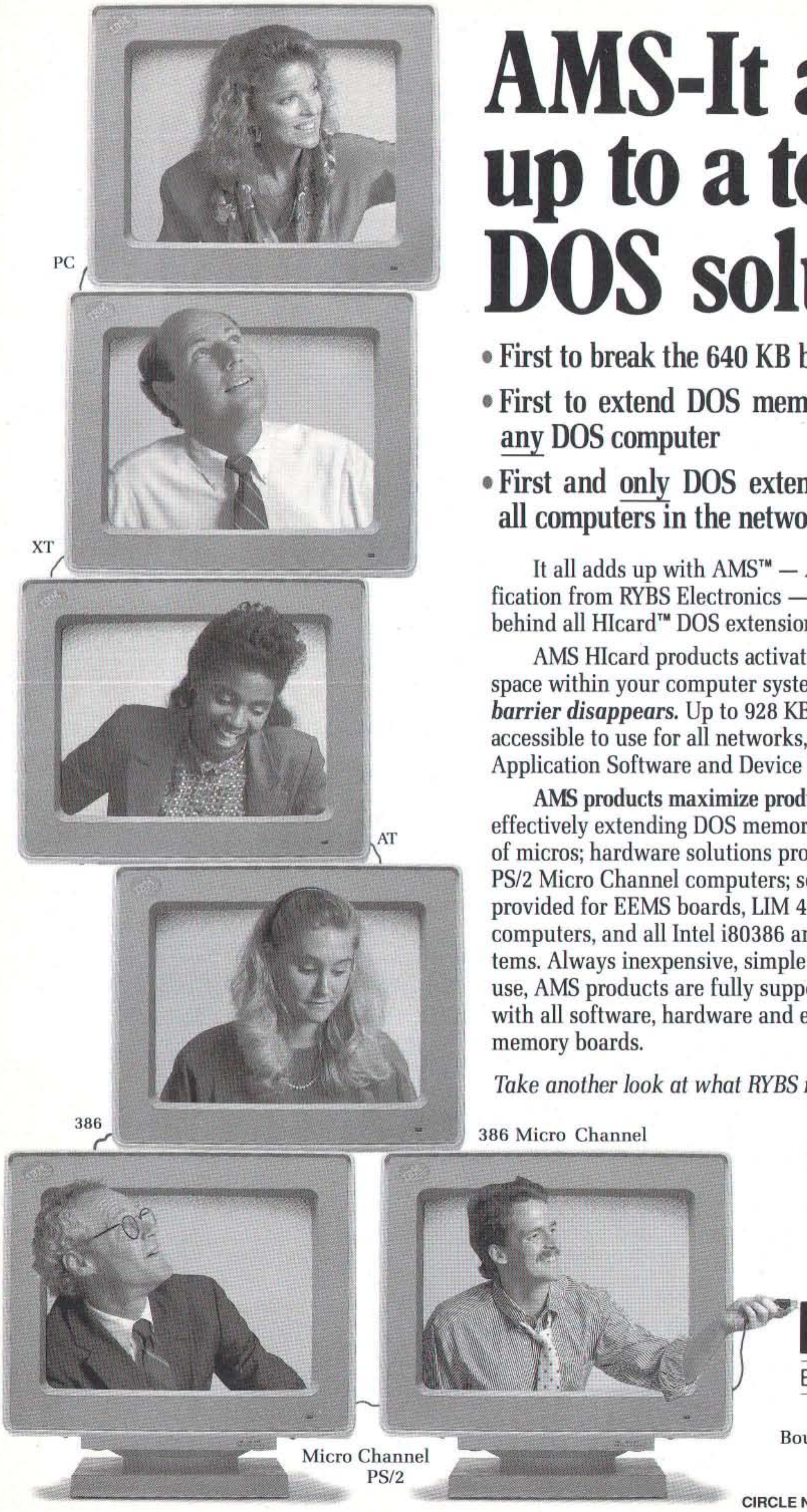
```
/* -------------------- filesort.c ----------------------- */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "csort.h"

/* sort a file:
 *     command line: filesort filename length (f1, l1, az1, ...)
 *     filename is the name of the input file
 *         length is the length of a record
 *         for each field:
 *             f1 is the field position relative to 1
 *             l1 is the field lengths
 *             az1 = A = ascending, D = descending
 */

static struct s_prm sp;
static void usage(void);

void main(int argc, char *argv[])
{
    int i, fct = 0;
    FILE *fpin, *fpout;
    char *buf;
    char filename[64];

    /* ------- get the file name from the command line ------ */
    if (argc-- > 1)
        strcpy(filename, argv++[1]);
    else
        usage();
    /* ----- get the record length from the command line ---- */
    if (argc-- > 1)
        sp.rc_len = atoi(argv++[1]);
    else
        usage();
    /* ----- get field definitions from the command line ---- */
    do  {
        if (argc < 4)
            usage();
        sp.s_fld[fct].f_pos = atoi(argv++[1]);
        sp.s_fld[fct].f_len = atoi(argv++[1]);
        sp.s_fld[fct].ad = *argv++[1];
        argc -= 3;
        fct++;
    } while (argc > 1);

    printf("\nFile: %s, length", filename, sp.rc_len);
    for (i = 0; i < fct; i++)
        printf("\nField %d: position %d, length %d, %s",
            i+1,
            sp.s_fld[i].f_pos,
            sp.s_fld[i].f_len,
            sp.s_fld[i].ad == 'd' ?
                            "descending" : "ascending");

    if ((fpin = fopen(filename, "rb")) == NULL) {
        printf("\nInput file not found");
        exit(1);
    }
    if ((buf = malloc(sp.rc_len)) == NULL ||
                init_sort(&sp) == -1)    {
        printf("\nInsufficient memory to sort");
        exit(1);
    }
    /* ------ sort the input records ------- */
     while (fread(buf, sp.rc_len, 1, fpin) == 1)
         sort(buf);
     sort(NULL);
     fclose(fpin);
     /* ----- retrieve the sorted output records ------ */
     fpout = fopen("SORTED.DAT", "wb");
     while ((buf = sort_op()) != NULL)
         fwrite(buf, sp.rc_len, 1, fpout);
    fclose(fpout);
    sort_stats();
    free(buf);
}

static void usage(void)
{
    printf("\nusage: filesort fname len {pos length ad...}");
    exit(1);
}
```

End Listing Two

Listing Three

```
/* ---------------------- csort.c ------------------------ */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "csort.h"

static struct s_prm *sp;  /* structure of sort parameters      */
static unsigned totrcd;   /* total records sorted              */
static int no_seq;        /* counts sequences                  */
static int no_seq1;
static unsigned bspace;   /* available buffer space            */
static int nrcds;         /* # of records in sort buffer       */
static int nrcds1;
static char *bf, *bf1;    /* points to sort buffer             */
static int inbf;          /* variable records in sort buffer */
static char **sptr;       /* -> array of buffer pointers       */
static char *init_sptr;   /* pointer to appropriated buffer  */
static int rcds_seq;      /* rcds / sequence in merge buffer */
static FILE *fp1, *fp2;   /* sort work file fds                */
static char fdname[15];   /* sort work name                    */
static char f2name[15];   /* sort work name                    */

static int comp(char **a, char **b);
static char *appr_mem(unsigned *h);
static FILE *wopen(char *name, int n);
static void dumpbuff(void);
static void merge(void);
static void prep_merge(void);

/* -------- initialize sort global variables----------  */
int init_sort(struct s_prm *prms)
{
    sp = prms;
    if ((bf = appr_mem(&bspace)) != NULL)    {
        nrcds1 = nrcds = bspace / (sp->rc_len + sizeof(char *));
        init_sptr = bf;
        sptr = (char **) bf;
        bf += nrcds * sizeof(char *);
        fp1 = fp2 = NULL;
        totrcd = no_seq = inbf = 0;
        return 0;
    }
    else
        return -1;
}

/* --------- Function to accept records to sort ------------ */
void sort(char *s_rcd)
{
    if (inbf == nrcds)  {   /* if the sort buffer is full */
        qsort(init_sptr, inbf,
            sizeof (char *), comp);
        if (s_rcd)  {   /* if there are more records to sort  */
            dumpbuff(); /* dump the buffer to a sort work file*/
            no_seq++;   /* count the sorted sequences         */
        }
    }
    if (s_rcd != NULL)  {
        /* --- this is a record to sort --- */
        totrcd++;
        /* --- put the rcd addr in the pointer array --- */
        *sptr = bf + inbf * sp->rc_len;
        inbf++;
        /* --- move the rcd to the buffer --- */
        memmove(*sptr, s_rcd, sp->rc_len);
        sptr++;                     /* point to next array entry */
    }
    else    {                 /* null pointer means no more rcds */
        if (inbf)   {         /* any records in the buffer?      */
            qsort(init_sptr, inbf,
                sizeof (char *), comp);
            if (no_seq)       /* if this isn't the only sequence*/
                dumpbuff();   /* dump the buffer to a work file */
            no_seq++;         /* count the sequence              */
        }
        no_seq1 = no_seq;
        if (no_seq > 1)    /* if there is more than 1 sequence */
            prep_merge();  /* prepare for the merge            */
    }
}

/* -------------- Prepare for the merge ----------------- */
static void prep_merge()
{
    int i;
    struct bp *rr;
    unsigned n_bfsz;

    memset(init_sptr, '\0', bspace);
    /* -------- merge buffer size ------ */
```

(continued on page 146)



AMS-It adds up to a total DOS solution.

- First to break the 640 KB barrier
- First to extend DOS memory in <u>any</u> DOS computer
- First and <u>only</u> DOS extension that makes all computers in the network 100% compatible

It all adds up with AMS™ — Advanced Memory Specification from RYBS Electronics — the exclusive software behind all HIcard™ DOS extension products.

AMS HIcard products activate the unused memory space within your computer system. ***The 640 KB DOS barrier disappears.*** Up to 928 KB of real DOS memory is accessible to use for all networks, CONFIG.SYS files, Application Software and Device Drivers.

AMS products maximize productivity and profit — effectively extending DOS memory for the complete line of micros; hardware solutions provided for PC, XT, AT and PS/2 Micro Channel computers; software only solutions provided for EEMS boards, LIM 4.0 boards, 286 NEAT computers, and all Intel i80386 and i80486 computer systems. Always inexpensive, simple to install and easy to use, AMS products are fully supported and compatible with all software, hardware and extended/expanded memory boards.

Take another look at what RYBS is offering for the '90s!

RYBS
ELECTRONICS

P.O. Box 4521
Boulder, Colorado 80306 USA
Tel: 303-444-6073
FAX: 303-449-9259

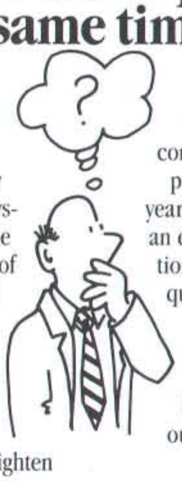
CIRCLE NO. 419 ON READER SERVICE CARD

Listing Three *(Listing continued, text begins on page 127.)*

```
    n_bfsz = bspace - no_seq * sizeof(struct bp);
    /* ------ # rcds/seq in merge buffer ------- */
    rcds_seq = n_bfsz / no_seq / sp->rc_len;
    if (rcds_seq < 2)    {
        /* ---- more sequence blocks than will fit in buffer,
                merge down ---- */
        fp2 = wopen(f2name, 2);      /* open a sort work file */
        while (rcds_seq < 2)     {
            FILE *hd;
            merge();                 /* binary merge */
            hd = fp1;                /* swap fds */
            fp1 = fp2;
            fp2 = hd;
            nrcds *= 2;
            /* ------ adjust number of sequences ------ */
            no_seq = (no_seq + 1) / 2;
            n_bfsz = bspace - no_seq * sizeof(struct bp);
            rcds_seq = n_bfsz / no_seq / sp->rc_len;
        }
    }
    bf1 = init_sptr;
    rr = (struct bp *) init_sptr;
    bf1 += no_seq * sizeof(struct bp);
    bf = bf1;

    /* fill the merge buffer with records from all sequences */

    for (i = 0; i < no_seq; i++)     {
        fseek(fp1, (long) i * ((long) nrcds * sp->rc_len),
                        SEEK_SET);
        /* ------ read them all at once ------ */
        fread(bf1, rcds_seq * sp->rc_len, 1, fp1);
        rr->rc = bf1;
        /* --- the last seq has fewer rcds than the rest --- */
        if (i == no_seq-1)   {
            if (totrcd % nrcds > rcds_seq)       {
                rr->rbuf = rcds_seq;
                rr->rdsk = (totrcd % nrcds) - rcds_seq;
            }
            else        {
                rr->rbuf = totrcd % nrcds;
                rr->rdsk = 0;
            }
        }
        else        {
            rr->rbuf = rcds_seq;
            rr->rdsk = nrcds - rcds_seq;
        }
        rr++;
        bf1 += rcds_seq * sp->rc_len;
    }
}
```

```
/* ------- Merge the work file down
    This is a binary merge of records from sequences
    in fp1 into fp2. ------ */
static void merge()
{
    int i;
    int needy, needx;   /* true = need a rcd from (x/y)   */
    int xcnt, ycnt;     /* # rcds left each sequence      */
    int x, y;           /* sequence counters              */
    long adx, ady;      /* sequence record disk addresses */

    /* --- the two sets of sequences are x and y ----- */
    fseek(fp2, 0L, SEEK_SET);
    for (i = 0; i < no_seq; i += 2)      {
        x = y = i;
        y++;
        ycnt =
            y == no_seq ? 0 : y == no_seq - 1 ?
            totrcd % nrcds : nrcds;
        xcnt = y == no_seq ? totrcd % nrcds : nrcds;
        adx = (long) x * (long) nrcds * sp->rc_len;
        ady = adx + (long) nrcds * sp ->rc_len;
        needy = needx = 1;
        while (xcnt || ycnt)     {
            if (needx && xcnt)  {   /* need a rcd from x? */
                fseek(fp1, adx, SEEK_SET);
                adx += (long) sp->rc_len;
                fread(init_sptr, sp->rc_len, 1, fp1);
                needx = 0;
            }
            if (needy && ycnt)  {   /* need a rcd from y? */
                fseek(fp1, ady, SEEK_SET);
                ady += sp->rc_len;
                fread(init_sptr+sp->rc_len, sp->rc_len, 1, fp1);
                needy = 0;
            }
            if (xcnt || ycnt)   {   /* if anything is left */
                /* ---- compare the two sequences --- */
                 if (!ycnt || (xcnt &&
                     (comp(&init_sptr, &init_sptr + sp->rc_len))
                        < 0))    {
                    /* ----- record from x is lower ---- */
                    fwrite(init_sptr, sp->rc_len, 1, fp2);
                    --xcnt;
                    needx = 1;
                 }
                 else if (ycnt)  {  /* record from y is lower */
                    fwrite(init_sptr+sp->rc_len,
                                  sp->rc_len, 1, fp2);
                    --ycnt;
                    needy = 1;
                 }
             }
        }
    }
}

/* -------- Dump the sort buffer to the work file ---------- */
static void dumpbuff()
{
    int i;

    if (fp1 == NULL)
        fp1 = wopen(fdname, 1);
    sptr = (char **) init_sptr;
    for (i = 0; i < inbf; i++)  {
        fwrite(*(sptr + i), sp->rc_len, 1, fp1);
        *(sptr + i) = 0;
    }
    inbf = 0;
}

/* --------------- Open a sort work file ------------------ */
static FILE *wopen(char *name, int n)
{
    FILE *fp;
    strcpy(name, "sortwork.000");
    name[strlen(name) - 1] += n;
    if ((fp =  fopen(name, "wb+")) == NULL) {
        printf("\nFile error");
        exit(1);
    }
    return fp;
}

/* --------- Function to get sorted records ---------------
This is called to get sorted records after the sort is done.
It returns pointers to each sorted record.
Each call to it returns one record.
When there are no more records, it returns NULL. ------ */

char *sort_op()
{
    int j = 0;
    int nrd, i, k, l;
    struct bp *rr;
    static int rl = 0;
    char *rtn;
    long ad, tr;

    sptr = (char **) init_sptr;
    if (no_seq < 2) {
        /* -- with only 1 sequence, no merge has been done -- */
        if (rl == totrcd)    {
            free(init_sptr);
            fp1 = fp2 = NULL;
            rl = 0;
            return NULL;
        }
        return *(sptr + rl++);
    }
```

How to be in two places at the same time...



If you're a manager responsible for designing, developing and introducing a new software product or system, then you know the feeling. The demands of software development make you wish you could be in two places at the same time. We can't put you in two places at one time, but we can lighten your work load. Our large, experienced software engineering staff is available for new product design and development, conversions and special projects. In the last six years, we have developed an extraordinary reputation for developing high quality, innovative software for our clients. But don't take our word for it, send for our portfolio and our free booklet today, **How to Select and Evaluate Software Development Companies.**

TPS Turning Point Software

230 Western Ave., Boston, MA 02134
617-782-4877

```
    rr = (struct bp *) init_sptr;
    for (i = 0; i < no_seq; i++)
        j |= (rr + i)->rbuf | (rr + i)->rdsk;

    /* -- j will be true if any sequence still has records - */
    if (!j)    {
        fclose(fp1);               /* none left */
        remove(fdname);
        if (fp2)    {
            fclose(fp2);
            remove(f2name);
        }
        free(init_sptr);
        fp1 = fp2 = NULL;
        r1 = 0;
        return NULL;
    }
    k = 0;

    /* --- find the sequence in the merge buffer
                        with the lowest record --- */
    for (i = 0; i < no_seq; i++)
        k = ((comp( &(rr + k)->rc, &(rr + i)->rc) < 0) ? k : i);

    /* --- k is an integer sequence number that offsets to the
        sequence with the lowest record ---- */

    (rr + k)->rbuf--;            /* decrement the rcd counter */
    rtn = (rr + k)->rc;          /* set the return pointer */
    (rr + k)->rc += sp->rc_len;
    if ((rr + k)->rbuf == 0)    {
        /* ---- the sequence got empty ---- */
        /* --- so get some more if there are any --- */
        rtn = bf + k * rcds_seq * sp->rc_len;
        memmove(rtn, (rr + k)->rc - sp->rc_len, sp->rc_len);
        (rr + k)->rc = rtn + sp->rc_len;
        if ((rr + k)->rdsk != 0)    {
            l = ((rcds_seq-1) < (rr+k)->rdsk) ?
                rcds_seq-1 : (rr+k)->rdsk;
            nrd = k == no_seq - 1 ? totrcd % nrcds : nrcds;
            tr = (long) ((k * nrcds + (nrd - (rr + k)->rdsk)));
            ad = tr * sp->rc_len;
            fseek(fp1, ad, SEEK_SET);
            fread(rtn + sp->rc_len, l * sp->rc_len, 1, fp1);
            (rr + k)->rbuf = l;
            (rr + k)->rdsk -= l;
        }
        else
            memset((rr + k)->rc, 127, sp->rc_len);
    }
    return rtn;
}
```

```
/* ------- Function to display sort stats -------- */
void sort_stats()
{
    printf("\n\n\nRecord Length = %d",sp->rc_len);
    printf("\n%d records sorted",totrcd);
    printf("\n%d sequence",no_seql);
    if (no_seql != 1)
        putchar('s');
    printf("\n%u characters of sort buffer", bspace);
    printf("\n%d records per buffer\n\n",nrcdsl);
}

/* ----- appropriate available memory ----- */
static char *appr_mem(unsigned *h)
{
    char *buff = NULL;

    *h = (unsigned) MOSTMEM + 1024;
    while (buff == NULL && *h > LEASTMEM)    {
        *h -= 1024;
        buff = malloc(*h);
    }
    return buff;
}

/* ------- compare function for sorting, merging -------- */
static int comp(char **a, char **b)
{
    int i, k;

    if (**a == 127 || **b == 127)
        return (int) **a - (int) **b;
    for (i = 0; i < NOFLDS; i++)    {
        if (sp->s_fld[i].f_pos == 0)
            break;
        if ((k = strnicmp((*a)+sp->s_fld[i].f_pos - 1,
                          (*b)+sp->s_fld[i].f_pos - 1,
                          sp->rc_len)) != 0)
            return (sp->s_fld[i].ad == 'd')?-k:k;
    }
    return 0;
}
```

End Listings

PANEL® *Plus* II

Screen Generator and User Interface Library for 'C'

PANEL *Plus* II features: Interactive screen design editor • C code generator • extensive library of user-interface functions • pop-up fields/windows • multiple-line fields • horizontal/vertical scrolling regions • scroll bar/mouse support • help boxes • custom validation • pull-down menus • complete **library source** included • support for popular **graphics libraries** • OS/2 DLL • **no royalties.**

Just released: **Utility Source Option** including source code for the screen layout editor and code generators, which allows **PANEL *Plus* II** to be ported to any suitable development platform. This gives you the freedom to choose and switch your development environment, with total screen and source code compatibilty . Also now supported: Microsoft **FORTRAN** under DOS and OS/2 (the DOS version requires the MS-C run-time library).

Panel *Plus* II can now run on any system with 'C' (but block-mode terminals are not supported). Cost per user: **$495** (including 640-page manual). Please call for quantity pricing, full version list, and free demo disk.

Roundhill Computer Systems Limited
1964 Richton Drive, Wheaton IL 60187
Telephone: **(708) 690 3737** • Fax: (708) 665 9841 • BIX: *join roundhill*
UK Head Office: POB 14 Marlborough, Wiltshire SN8 1LG • Tel: **0672 84535** • Fax: 0672 84525

AccSys™ for C dBASE

- Set of C libraries for easy access to dBASE data and index files.
- No need to concern yourself with internal files format.
- Increased performance shortens processing time.
- Total control over data memo, MDX and NDX index files.
- Network and single-user versions available.

Copia International Ltd.
1964 Richton Drive, Wheaton, Illinois 60187
708/682-8898 FAX: 708/665-9841
Instant info. via FaxFacts: 617/740-0025 #1625 007

Listing One *(Text begins on page 135.)*

```
{--------------------------------------------------------}
{                       TIMEDATE                         }
{                                                        }
{ A Time-and-date stamp object for Turbo Pascal 5.5      }
{                                                        }
{                          by Jeff Duntemann             }
{                          Last update 12/23/89          }
{                                                        }
{ NOTE: This unit should be good until December 31,      }
{ 2043, when the long integer time/date stamp turns      }
{ negative.  HOWEVER, the Zeller's Congruence            }
{ algorithm shown here fails at the end of the 20th      }
{ century.  I should be able to figure out the fix       }
{ by then...                                             }
{--------------------------------------------------------}

UNIT TimeDate;

INTERFACE

USES DOS;

TYPE
  String9  = STRING[9];
  String20 = STRING[20];
  String50 = STRING[50];

  When =
    OBJECT
      WhenStamp      : LongInt;        { Combined time/date stamp }
      TimeString     : String9;        { i.e., "12:45a"           }
      Hours,Minutes,Seconds : Word;    { Seconds is always even!  }
      DateString     : String20;       { i.e., "06/29/89"         }
      LongDateString : String50;       { i.e., "Thursday, June 29, 1989" }
      Year,Month,Day : Word;
      DayOfWeek      : Integer;        { 0=Sunday, 1=Monday, etc. }
      FUNCTION GetTimeStamp : Word;    { Returns DOS-format time stamp }
      FUNCTION GetDateStamp : Word;    { Returns DOS-format date dtamp }
      PROCEDURE PutNow;
      PROCEDURE PutWhenStamp(NewWhen  : LongInt);
      PROCEDURE PutTimeStamp(NewStamp : Word);
      PROCEDURE PutDateStamp(NewStamp : Word);
      PROCEDURE PutNewDate(NewYear,NewMonth,NewDay : Word);
      PROCEDURE PutNewTime(NewHours,NewMinutes,NewSeconds : Word);
    END;


IMPLEMENTATION

{ Keep in mind that all this stuff is PRIVATE to the unit! }

CONST
  MonthTags : ARRAY [1..12] of String9 =
    ('January','February','March','April','May','June','July',
     'August','September','October','November','December');
  DayTags   : ARRAY [0..6] OF String9 =
    ('Sunday','Monday','Tuesday','Wednesday',
     'Thursday','Friday','Saturday');

TYPE
  WhenUnion =
    RECORD
      TimePart : Word;
      DatePart : Word;
    END;

VAR
  Templ : String50;
  Dummy : Word;

{ Some utility routines private to this unit: }

FUNCTION CalcTimeStamp(Hours,Minutes,Seconds : Word) : Word;

BEGIN
  CalcTimeStamp := (Hours SHL 11) OR (Minutes SHL 5) OR (Seconds SHR 1);
END;


FUNCTION CalcDateStamp(Year,Month,Day : Word) : Word;

BEGIN
  CalcDateStamp := ((Year - 1980) SHL 9) OR (Month SHL 5) OR Day;
END;


PROCEDURE CalcTimeString(VAR TimeString : String9;
                         Hours,Minutes,Seconds : Word);

VAR
  Templ,Temp2 : String9;
  AMPM        : Char;
  I           : Integer;

BEGIN
  I := Hours;
  IF Hours = 0 THEN I := 12;   { "0" hours = 12am }
  IF Hours > 12 THEN I := Hours - 12;
  IF Hours > 11 THEN AMPM := 'p' ELSE AMPM := 'a';
  Str(I:2,Templ); Str(Minutes,Temp2);
  IF Length(Temp2) < 2 THEN Temp2 := '0' + Temp2;
  TimeString := Templ + ':' + Temp2 + AMPM;
END;


PROCEDURE CalcDateString(VAR DateString : String20;
                         Year,Month,Day : Word);
BEGIN
  Str(Month,DateString);
  Str(Day,Templ);
  DateString := DateString + '/' + Templ;
  Str(Year,Templ);
  DateString := DateString + '/' + Copy(Templ,3,2);
END;


PROCEDURE CalcLongDateString(VAR LongdateString : String50;
                             Year,Month,Date,DayOfWeek : Word);
VAR
  Templ : String9;

BEGIN
  LongDateString := DayTags[DayOfWeek] + ', ';
  Str(Date,Templ);
  LongDateString := LongDateString +
    MonthTags[Month] + ' ' + Templ + ', ';
  Str(Year,Templ);
  LongDateString := LongDateString + Templ;
END;


{---------------------------------------------------------------------}
{ This calculates a day of the week figure, where 0=Sunday, 1=Monday, }
{ and so on, given the year, month, and day.  The year may be passed  }
{ as either "1989" or "89" but *not* as 1980-relative, or "9".  Also  }
{ note that this particular algorithm turns into a pumpkin in 2000.   }
{ BTW, don't ask me to explain how this crazy thing works.  I haven't }
{ the foggiest notion.  If I ever meet Mr. Zeller, I'll ask him.      }
{---------------------------------------------------------------------}

FUNCTION CalcDayOfWeek(Year,Month,Day : Word) : Integer;

VAR
  Century,Leftovers,Holder : Integer;

BEGIN
  { First test for error conditions on input values: }
  IF (Year < 0)  OR
     (Month < 1) OR (Month > 12) OR
     (Day < 1)   OR (Day > 31) THEN
      CalcDayOfWeek := -1  { Return -1 to indicate an error }
  ELSE
    { Do the Zeller's Congruence calculation: }
    BEGIN
      IF Year < 100 THEN Inc(Year,1900);
      Dec(Month,2);
      IF (Month < 1) OR (Month > 10) THEN
        BEGIN
          Dec(Year,1);
         Inc(Month,12);
        END;
      Century   := Year DIV 100;
      Leftovers := Year MOD 100;
      Holder    := (Trunc(Int(2.6 * Month - 0.2)) + Day +
                   Leftovers + (Leftovers DIV 4) +
                   (Century DIV 4) - Century - Century) MOD 7;
      IF Holder < 0 THEN
        Inc(Holder,7);
      CalcDayOfWeek := Holder;
    END;
END;


{**************************************}
{ Method implementations for type When: }
{**************************************}


{---------------------------------------------------------------------}
{ There will be many times when an individual date or time stamp will }
{ be much more useful than a combined time/date stamp.  These simple  }
{ functions return the appropriate half of the combined long integer  }
{ time/date stamp without incurring any calculation overhead.  It's   }
{ done with a simple value typecast:                                  }
{---------------------------------------------------------------------}

FUNCTION When.GetTimeStamp : Word;

BEGIN
  GetTimeStamp := WhenUnion(WhenStamp).TimePart;
END;


FUNCTION When.GetDateStamp : Word;

BEGIN
  GetDateStamp := WhenUnion(WhenStamp).DatePart;
END;


{---------------------------------------------------------------------}
{ To fill a When record with the current time and date as maintained  }
{ by the system clock, execute this method:                           }
{---------------------------------------------------------------------}

PROCEDURE When.PutNow;

BEGIN
  { Get current clock time.  Note that we ignore hundredths figure: }
  GetTime(Hours,Minutes,Seconds,Dummy);
  { Calculate a new time stamp and update object fields: }
  PutTimeStamp(CalcTimeStamp(Hours,Minutes,Seconds));
  GetDate(Year,Month,Day,Dummy); { Get current clock date }
  { Calculate a new date stamp and update object fields: }
  PutDateStamp(CalcDateStamp(Year,Month,Day));
END;


{---------------------------------------------------------------------}
{ This method allows us to apply a whole long integer time/date stamp }
{ such as that returned by the DOS unit's GetFTime procedure to the   }
{ When object.  The object divides the stamp into time and date          }
{ portions and recalculates all other fields in the object.              }
{------------------------------------------------------------------------}

PROCEDURE When.PutWhenStamp(NewWhen  : LongInt);

BEGIN
  WhenStamp := NewWhen;
  { We've actually updated the stamp proper, but we use the two }
  { "put" routines for time and date to generate the individual }
  { field and string representation forms of the time and date. }
  { I know that the "put" routines also update the long integer }
  { stamp, but while unnecessary it does no harm.               }
  PutTimeStamp(WhenUnion(WhenStamp).TimePart);
  PutDateStamp(WhenUnion(WhenStamp).DatePart);
END;


{------------------------------------------------------------------------}
{ We can choose to update only the time stamp, and the object will       }
{ recalculate only its time-related fields.                              }
{------------------------------------------------------------------------}

PROCEDURE When.PutTimeStamp(NewStamp : Word);

BEGIN
  WhenUnion(WhenStamp).TimePart := NewStamp;
  { The time stamp is actually a bitfield, and all this shifting left }
  { and right is just extracting the individual fields from the stamp:}
  Hours := NewStamp SHR 11;
  Minutes := (NewStamp SHR 5) AND $003F;
  Seconds := (NewStamp SHL 1) AND $001F;
  { Derive a string version of the time: }
  CalcTimeString(TimeString,Hours,Minutes,Seconds);
END;


{------------------------------------------------------------------------}
{ Or, we can choose to update only the date stamp, and the object        }
{ will then recalculate only its date-related fields.                    }
{------------------------------------------------------------------------}

PROCEDURE When.PutDateStamp(NewStamp : Word);

BEGIN
  WhenUnion(WhenStamp).DatePart := NewStamp;
  { Again, the date stamp is a bit field and we shift the values out  }
  { of it: }
  Year := (NewStamp SHR 9) + 1980;
  Month := (NewStamp SHR 5) AND $000F;
  Day := NewStamp AND $001F;
  { Calculate the day of the week value using Zeller's Congruence:    }
  DayOfWeek := CalcDayOfWeek(Year,Month,Day);
  { Calculate the short string version of the date; as in "06/29/89": }
  CalcDateString(DateString,Year,Month,Day);
  { Calculate a long version, as in "Thursday, June 29, 1989": }
  CalcLongDateString(LongdateString,Year,Month,Day,DayOfWeek);
END;


PROCEDURE When.PutNewDate(NewYear,NewMonth,NewDay : Word);

BEGIN
  { The "boss" field is the date stamp.  Everything else is figured }
  { from the stamp, so first generate a new date stamp, and then    }
  { (odd as it may seem) regenerate everything else, *including*    }
  { the Year, Month, and Day fields: }
  PutDateStamp(CalcDateStamp(NewYear,NewMonth,NewDay));
  { Calculate the short string version of the date; as in "06/29/89": }
  CalcDateString(DateString,Year,Month,Day);
  { Calculate a long version, as in "Thursday, June 29, 1989": }
  CalcLongDateString(LongdateString,Year,Month,Day,DayOfWeek);
END;


PROCEDURE When.PutNewTime(NewHours,NewMinutes,NewSeconds : Word);

BEGIN
  { The "boss" field is the time stamp.  Everything else is figured }
  { from the stamp, so first generate a new time stamp, and then    }
  { (odd as it may seem) regenerate everything else, *including*    }
  { the Hours, Minutes, and Seconds fields: }
  PutTimeStamp(CalcTimeStamp(NewHours,NewMinutes,NewSeconds));
  { Derive the string version of the time: }
  CalcTimeString(TimeString,Hours,Minutes,Seconds);
END;
```

End Listing One

Listing Two

```
PROGRAM TimeTest;

USES Crt,TimeDate;

VAR
  Now : When;

BEGIN
  Write('At the tone, it will be exactly ');
  Delay(1000);
  Now.PutNow;
  Sound(1000); Delay(100); NoSound;
  WITH Now DO Writeln(TimeString,'m on ',LongDateString,'.');
  Readln
END.
```

End Listings

FINALLY
THE IDEAL "C" SEMINAR!

Within one week, you can increase the value of your employees by adding "C" to your company's programming language vocabulary!

CONVENIENT COST-EFFICIENT "C" TRAINING

Thousands of people have been trained and they have given CLA courses consistently superior reviews with our competitively priced courses. Become proficient in ANSI "C" in one week.

FLEXIBLE SCHEDULING

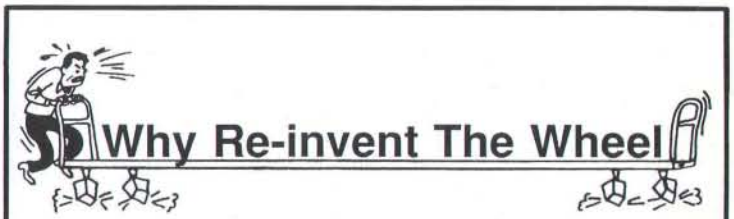
Public classes are offered monthly in our Collingswood office, convenient to Philadelphia. Classes can also be scheduled at your company site.

FOR FAST INFORMATION
CALL TOLL FREE
1-800-842-9667
and we will FAX or mail you the latest brochures

CLA INC
COMPUTER LANGUAGE ARTS

704 Haddon Avenue; Collingswood, NJ 08108

CIRCLE NO. 388 ON READER SERVICE CARD



Looking for a powerful, feature packed text editor to be included into your application? Now available:

TE Developer's Kit

TE Developer's Kit is the most cost effective way of incorporating text editing features into your application. The kit includes complete source code of the editor, library routines and the executable file. TE Developers Kit supports Microsoft and Borland 'C' compilers.

The Features Include:

• Complete full screen editor. • Multiple files/windows. • Word wrap and printing functions. • Support for files larger than available memory. • Reconfigurable Keyboard commands and screen colors. • Keystroke macros. • Undo. • Columnar block commands. • Search/replace. •Autosave.

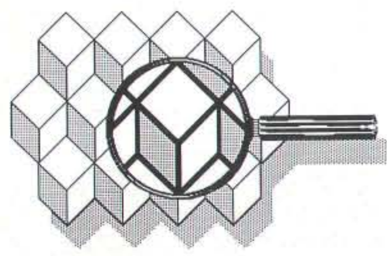
The kit includes TES small editor routine. TES provides scrolling functions, word-wrap and block commands. An application can call this routine without any modifications to the routine. Your application specifies window co-ordinates, maximum file size, and an input buffer or file.

The Kit is most favorable priced at $139.00

To Place an Order, Call

Sub Systems, Inc. • 1-800-447-6819
159 Main Street, #8C, Stoneham, MA 02180 • 1-617-438-8901

CIRCLE NO. 264 ON READER SERVICE CARD



Actor 2.0, an object-oriented development environment for Microsoft Windows applications, has been announced by **The Whitewater Group**. Version 2.0 boasts an automatic object swapping system that swaps static objects and code out to disk, which allows developers to break the 640K barrier of MS-DOS and create applications that are larger than 1 Mbyte in size. It does this by means of an LRU (least-recently used) algorithm, which ages unused objects and sends the old ones out to disk.

Actor 2.0 also has additional object-oriented features, new commands, and improved support for C. Whitewater spokesman Zack Urlocker told *DDJ* that "programmers wanted increased compatibility with C so we've added the ability to pass C structures and combine primitives in C and assembly language. This means you can get directly into the low-level structure of objects." 2.0 runs on any PC or PS/2 (or compatibles) under Microsoft Windows, and requires 640K of memory, a hard disk, a graphics card of any resolution, a mouse, and Microsoft Windows 2.x or later. The cost is $695, but registered users of previous versions can upgrade for $149. Reader service no. 20.
The Whitewater Group
600 Davis St.
Evanston, IL 60201
708-328-3800

Version 2.00 of Top Speed Modula-2 has been announced by **JPI**. This version works with the multilanguage development recently released with TopSpeed C, which automatically selects the appropriate compiler for the source file it needs to compile, and includes nine editing windows, 500K per file, and a hypertext help system. New compiler options allow interfacing of C libraries and functions with Modula-2 programs, and vice versa. The optimizing code generator is featured in the new family of TopSpeed languages (including a TopSpeed Pascal, which has also been announced), common to all the languages.

TopSpeed Modula-2, Version 2.00, has multiple memory model support, and new keywords and syntax extensions allow object-oriented programming. A new keyword CLASS allows a RECORD-like structure to include PROCEDURE declarations, hide data, specify inheritance and VIRTUAL procedures. And syntax extensions allow the name of a variable of a CLASS type to qualify the name of a procedure.

Included now in all TopSpeed compilers is the Visual Interactive Debugger (VID), which is a multilanguage source-level debugger that features on-screen display of all breakpoint traps, single-step operation, and full source code inspection. TopSpeed Modula-2 is available for DOS ($395 for extended edition) and OS/2 ($495 for extended edition). Reader service no. 33.
Jensen & Partners International
1101 San Antonio Rd., Ste. 301
Mountain View, CA 94043
415-967-3200

A software development tool that gives the 80386 processor the same level of protection available in protected operating systems has been announced by **Nu-Mega**. Bounds-Checker automatically detects out-of-bounds accesses by an application program, using the symbolic information created by the Microsoft C 5.X and 6.0 compilers to show you the exact source line causing the out-of-bounds access. Nu-Mega claims that Bounds-Checker cuts steps in the development process and eliminates the need to debug, and that it prevents serious side effects of subtle overwrites that may not be particularly dangerous until the program is in the field.

The Bounds-Checker provides real-time memory protection, differentiates between code and data, protects program code and all memory outside your program, and prevents the system software from corrupting your program — it can determine if a TSR or other program is trouncing your program. Sells for $249. Reader service no. 35.
Nu-Mega Technologies
P.O. Box 7607
Nashua, NH 03060-7607
603-888-2386

Borland International has announced Version 2.0 of its Turbo Debugger, which includes a toolkit that features the new Turbo Profiler. The profiler measures where in your program time is spent, how many times a line is executed, how many times a routine is called and by what, and which files are accessed most often and for how long. It also tracks the use of resources such as processor time, disk access, keyboard input, printer output, and interrupt activity.

The Turbo Profiler graphically displays where your program is spending its time, telling you which parts of the program are used most often and may need optimization or rewriting, and which parts are used so little that you needn't bother tightening them. An optimizing compiler generates code for the program you give it.

Multiple overlapping windows, icons, mouse support, and context-sensitive on-line help are included in the user interface. The Profiler works with Turbo Pascal 5.0, Turbo C 2.0, and Turbo Assembler 1.0, and any later versions of these compilers. Also supports CodeView and .MAP file debug formats. For IBM PCs and compatibles operating PC-DOS (MS-DOS) 2.0 or later. Requires 384K (256K for Turbo Assembler). The toolkit retails for $149.95. Reader service no. 34.
Borland International
1800 Green Hills Rd.
Scotts Valley, CA 95066-0001
408-438-8400

Version 4.0 of the LALR compiler construction toolkit can now be purchased from **LALR Research**. The toolkit includes the parser generator, LALR, and a scanner generator, DFA, as well as various source code modules, such as a main program, a screener, parser skeleton, and scanner skeleton. *DDJ* spoke with the developer, Paul Mann, who said that the LALR compiler is "an advanced compiler construction tool that goes beyond YACC. It features extended BNF notation, automatic creation of abstract syntax trees, and a high-speed scanner generator. It's well suited for developing compilers, translators, and interpreters for computer languages."

A BNF grammar describes the statements of the language as input to the LALR, and describes the symbols as input to the DFA. The LALR parser generator provides automatic error recovery, and handles large grammars such as Fortran and Cobol. The company claims that the DFA scanner generator produces high-speed deterministic finite automatons that run about four times faster than LEX scanners. The source code output is compatible with Turbo C, Microsoft C, Watcom C, among others. The price is $495. Reader service no. 26.
LALR Research
1892 Burnt Mill Rd.
Tustin, CA 92680
714-832-2274

(continued on page 157)

THE ADVERTISER INDEX

RS#	Advertiser Name	Page #
82	Abacus Software	62
58	Abraxas Software Inc./Conley Computing	64
75	Abraxas Software, Inc.	55
312	Arrowhead Software	130
*	Aspen Scientific	132
325	Aspen Technologies Inc.	35
*	AT&T Information Systems	49
295	Atron	C3
188	Attachmate Corporation	59
89	Austin Code Works, The	117
72	Basetwo Development	131
407	Black & White International	63
91	Blaise Computing Inc.	2
159	Blaise Computing Inc.	122
95	Borland International	C4
406	Bradford Business Systems	85
247	Burton Systems	158
350	C Source Inc.	65B
99	C Users Journal, The	111
424	California Scientific Software	113
405	Computer Connection, The	158
388	Computer Lanuage Arts Inc	151
413	Computer Systems Advisers, Inc.	33
110	Courseware Applications, Inc.	142
*	Creative Programming	76
310	Datalight	54
376	DigiBoard	57
115	Digitalk, Inc.	90, 91
422	Direct Technology	61
117	Disk Software	108
387	Dovetail Communications Technologies Inc	24
417	Eikon Systems	141
128	FairCom Corporation	97
285	Genus Microprogramming	65G
*	Gimpel Software	26
225	Glenco Engineering Inc.	60
66	Graphic Software Systems	95
*	Greenleaf Software	123
351	Guidelines Software, Inc.	53
137	Harvard Softworks	79
284	ICS/Integrated Computer Solutions	121
119	Intek Integration Technologies	140
303	Introl Corp	50
304	Island Systems	4
286	Ithaca Street Software Inc	18
*	JPI Jensen & Partners International	29
275	JYACC Inc.	39
259	Kadak Products Ltd.	102
118	KBM	88
*	Klondike Software	30
340	Knowledge Dynamics Corporation	115
142	Lattice, Inc.	65D
344	Lloyd Bush, Inc.	24
131	Logitech, Inc.	51
144	Lugaru Software Ltd.	94
148	Magna Carta Software	85
289	Mansfield Software Group	140
*	Mark Williams Company	37
412	Marlton Career Connection	111
86	McCarty Associates	20
156	Metagraphics Software Corporation	103
158	MetaWare Incorporated	31
429	Metheus Corporation	67
430	Micro Data Base Systems Inc.	105
*	Microsoft	C2, 1
*	Microsoft University	83
181	Microsystems Software, Inc.	58
166	Mix Software	134
347	Nanosoft	143
171	Nantucket Corporation	52
170	NeuralWare, Inc.	19
113	Novell	5
174	Nu-Mega Technologies	13
178	Oakland Group, Inc.	65F
41	Ontologic Inc.	99
163	OPENetwork	81
265	Oregon Software	54
321	Popkin Software & Systems, Inc.	21
192	Programmer's Connection	159
195	Programmer's Paradise	10, 11
197	Programmers Shop, The	106, 107
385	Programmers Warehouse	87
199	Programmer's Wholesaler	120
210	Protech Marketing Inc.	128
326	Quantasm Corporation	102
203	Quantum Software Systems Ltd.	7
204	Quarterdeck Office Systems	92
431	Que Corporation	124
427	Quickbyte Software, Inc.	43
420	Raima Corporation	129
287	Raima Corporation	41
*	Roundhill	147
419	Rybs Electronics, Inc.	145
425	SAIC	74
190	Sage Software	22, 23
191	Sage Software	9
42	Santa Cruz Operations, The	15
208	Scientific Endeavors	96
212	SemiDisk Systems	101
213	Sequiter Software Inc	139
46	Silicon Valley Software	71
423	Software Security, Inc.	45
229	Solution Systems	136
325	Solution Systems	118
232	Stony Brook Software	44
264	Sub Systems, Inc.	151
237	Symmetry Group, The	116
219	T&T Computer Products	104
411	Tall Tree Systems	158
426	Terra	96
239	Tool Makers, The	141
240	TurboPower Software	137
*	Turning Point Software	146
245	V Communications, Inc.	133
414	Venn, Inc	108
*	Vermont Creative Software	47
104	Vestronix Inc.	126
416	Wenham Software Company	65
254	Whitewater Group, The	89
256	Xian Corporation	101
257	Zortech	25
258	Zortech	27

* The advertiser prefers to be contacted by phone; consult ad.

Advertising Sales Offices

Account Managers

Charlie Shively	Northwest	(415) 366-3600
Diane Fuller	Northern California	(415) 366-3600
Tim Trickett	Midwest	(415) 366-3600
Cindy Zuck	Northeast	(914) 762-9552
Cheri Blum	Southeast/Southwest	(714) 761-0294
Mary Kay Hoal	Northeast	(415) 366-3600

Programmer's MARKETPLACE

How to place your ad in the Programmer's Marketplace:

Carefully type your message or send camera-ready art. Each ad is 10 x 13 picas (15/8 x 23/16 inches). All ads must be prepaid. We accept checks, money orders, Visa, Mastercard, and American Express. Send your ad to: ***Programmer's Marketplace,*** *Dr. Dobb's Journal,* 501 Galveston Drive, Redwood City, CA 94063. For more information, contact: Glynn Mansfield at (415) 366-3600.

FIFTH

A revolutionary 32-bit implementation of a Forth-83 compiler, coupled with a high-productivity programming environment • truly compiled code • context-sensitive help • source-level debugger • stand-alone application generation Used to develop the first commercial PostScript™ clone.

SOFTWARE CONSTRUCTION COMPANY
2900B Longmire • College Station • Texas • 77840
(409) 696-5432

CIRCLE NO. 351 ON READER SERVICE CARD

The $25 Network

Try the 1st truly low cost LAN

- **Connect 2 or 3 PCs, XTs, ATs, PS/2s, 386s**
- **Uses serial ports and 5 wire null modem cable**
- **Runs at 115K baud, up to 90 feet**
- **Runs in background, totally transparent**
- **Share any device, any file, any time**
- **Needs only 14K of ram**
- **Version 2.3i** • **OVER 15,000 SOLD**

Skeptical? We make believers!

Information Modes
P.O. Drawer F, Denton, TX 76202
817-387-3339 Technical, 7 days
1-800-628-7992 Orders

CIRCLE NO. 352 ON READER SERVICE CARD

PRINTED GRAPHICS WITH TURBO PASCAL

GRAPHLINK is the powerful printer software that does for your printer what TURBO Pascal's BGI graphics does for your screen–at the printer's resolution!
GRAPHLINK emulates TURBO'S BGI commands, so adding (high-resolution) printer graphics to your program is a snap!
Supports HP LaserJet and DeskJet, Epson, IBM, Toshiba, NEC and other printers.
Only $25 for single-user version, or $250 for Graphlink Professional, which includes unlimited distribution license.
Shipping $5 / $10 / $15 for USA / Canada / other.

VISITECH SOFTWARE
D5 3807 Ridgewood Ct. Pittsburgh, Pa 15239
(412) 733-4775

CIRCLE NO. 353 ON READER SERVICE CARD

GRAPHIC TOOLS LIBRARY

XGLIB: Blazing Fast. User coord., thick lines and arcs. Polygon Engine. Figure drawing. Splines. Fill. Text scale, rotate, align. Keyb, Mouse. Plots, charts and presentation graphics with a difference. Screen print. **$99.**
PC VDI: Display and Printed graphics. Draw at printer resolution. Outline fillable font factory. Pan, Scroll.Includes XGLIB. All GKS Draw. **NO ROYALTIES.** Take **NOVA PRINVIEW** test. Compare Quality. **$395.**
SCANPRO: Image tools library. Image Capture. Fast Bitmap Graphics. Bitblt. Cut, Copy, Paste & more. Animation. Image Scale up-down, Rotate, Mirror, Tile fill and Window Scroll. Image data base. Text and Line drawing. Fonts. Virtual Bitmaps. Pop-up's. Scaled Print/plot. Keyboard, "better" mouse & Video paging. EMS support & 149 functions. Multiple image file formats. A better package for .PCX file handling. Compare Quality. **$149.**
PCXIO: Source Library available in "C"/ Assembler for reading/writing and display of .PCX files. **$295.**
ALL PRODUCTS: ANSI compatible. Include resident program for screen print. Herc. to Super VGA. Most **"C", PASCAL, FORTRAN, MS QuickBASIC and BASIC.**

NOVA INC.
2500 W. Higgins Road
Suite 1144
Hoffman Estates, IL 60195
708-882-4111 FAX: 708-882-4173

CIRCLE NO. 356 ON READER SERVICE CARD

FREE TRIAL

Pinnacle File Manager

We guarantee immediate productivity gains from using our C file manager library. Simple table-and-column view of data with advanced B-Tree indexing and hi-speed cache frees your brain to solve application-specific problems.

Vermont Database Corp.
RR3 Box 6516
Stowe, VT 05672
802-253-4437

CIRCLE NO. 354 ON READER SERVICE CARD

B.Sc. & M.S. in COMPUTER SCIENCE

The American Institute for Computer Sciences offers an in depth correspondence program to earn your Bachelor of Science and Master of Science degrees in Computer Science at home. BSc. subjects covered are: MS/DOS, BASIC, PASCAL, C, Data File Processing, Data Structures & Operating systems. MS program includes subjects in Software Engineering and Artificial Intelligence.

American Inst. for COMPUTER SCIENCES
1704-DJ 11TH Avenue South
Birmingham, AL 35205
(205) 933-0339

CIRCLE NO. 355 ON READER SERVICE CARD

Get Inside!

NEW RELEASE!

The best selling PC software performance tool is now better than ever with source line timing, caller timing and arbitrary event timing – all with microsecond accuracy and without source modification. The expanded DOS analysis mode identifies each DOS function call and potential I/O bottlenecks.

The list of supported compilers:

Turbo C	Microsoft BASIC
Microsoft C/Quick C	QuickBASIC
Microsoft FORTRAN	WATCOM C
Logitech Modula 2	JPI TopSpeed Modula 2
Zortech C/C + +	Lattice C
Turbo Pascal	

$125.00 EACH

30-day guarantee VISA/MC/COD
Call today for a free brochure and the latest list of supported compilers and operating environments.

Paradigm Systems
PO Box 152, Milford, Massachusetts 01757
(800)537-5043 In MA: (508)478-0499

CIRCLE NO. 357 ON READER SERVICE CARD

SOURCE LEVEL DEBUGGER FOR THE IBM ARTIC CARD.

ATTENTION ARTIC CARD PROGRAMMERS. Finally, a source level debugger. SARTICDB allows source level debugging for multiple tasks, and on multiple cards. Independent of system unit operating system. (i.e. DOS, OS2 etc.) No more guess work when finding bugs. Exterminate bugs now with SARTICDB. Call for more information.

ATLANTIC FIRMWARE
(609) 795-9651

CIRCLE NO. 358 ON READER SERVICE CARD

68000 *HIGH SPEED* CROSS-ASSEMBLER

Macros/ Conditional Assembly

68000/68010 code on a PC Compatible

4 MB of source code assembled per minute

S-Records, Binary, or Split Binary

SOFTWARE CONSTRUCTION COMPANY, INC.
2900B Longmire
College Station
Texas 77840

Phone (409) 696-5432

CIRCLE NO. 359 ON READER SERVICE CARD

B.S. DEGREES

. . . from fully-accredited Colleges. We help Computer Professionals avoid years of unnecessary class work. Get the job offers and recognition you deserve. Phone UDA for more information and our free booklet "Career Tactics and Strategies."

University Degree Advisory
☎ 800-765-7272

CIRCLE NO. 360 ON READER SERVICE CARD

DR. MD™

(a run time Memory Debugger)

FINALLY, a memory debugger for C that
- detects memory overwrites immediately
- catches invalid calls to free()
- reports dangling pointers
- displays the line # and file of the bug
- source code to fit your environment
- no extra hardware, no heap walking
- is a small library and is easy to use

BEST OF ALL, IT'S ONLY ... $59.95
(Plus $3.00 shipping & handling)
(Colorado residents add 3% for sales tax)
Send check or money order, for contact for free information:

TUITS COMPSULT, INC.
411 N. Shields
Fort Collins, CO 80521 (303) 224-9070

CIRCLE NO. 361 ON READER SERVICE CARD

Programmer's MARKETPLACE

How to place your ad in the Programmer's Marketplace:
Carefully type your message or send camera-ready art. Each ad is 10 x 13 picas (15/8 x 23/16 inches). All ads must be prepaid. We accept checks, money orders, Visa, Mastercard, and American Express. Send your ad to:
Programmer's Marketplace, *Dr. Dobb's Journal,* 501 Galveston Drive, Redwood City, CA 94063.
For more information, contact: Glynn Mansfield at (415) 366-3600.

FAST SCREEN MANAGER

For BASIC, C, FORTRAN, COBOL, or ASM. Menu, Data Entry, Pop-up and Pull Down Window support for all programmers. TSR or Linkable. 100 + page manual. 60-day, money-back guarantee. No royalties, Demo Available From $99.
VISA/MC.

The West Chester Group Inc.
P.O. Box 1304
West Chester, Pa 19380
(215) 644-4206 (800) 525-9244

CIRCLE NO. 362 ON READER SERVICE CARD

DRUMA FORTH-83
Break the 64k memory barrier

Powerful, attractively priced. '83 Standard
- 1Mb plus automated memory management
- On-line documentation, ASCII/block file
- Others: windows, modules, profiler-debugger
- Full OS interface & a lot more
- IBM PC/XT/AT and all compatibles

FREE learn/utility disks offered: INQUIRE

Druma Inc.
6448 Highway 290E, E103, Austin, Texas 78723
Orders: (512) 323-0403 BBoard: 512-323-2402

CIRCLE NO. 364 ON READER SERVICE CARD

MODULA-2 TOOLBOX

This library brings the professional touch to yor programs and helps you overcome difficult tasks! Features: Create back-up routines with the file utilities provided - Redirect I/O to ANY device - Stacks & queues only limited by memory - Virtual Screen & windowing systems - AUTO-DETECTS video adapter type (MDA, CGA, EGA, VGA) - User defined menus - RANDOM NUMBERS from within defined limits - LARGE assortment of math functions - Disk sector, FAT table & directory manipulation, AND MORE! Why waste time creating these tools? NO ROYALTIES! Call today to order your copy and give your programs the TOUCH. Object only: $99. Source Code: $299. *VISA & MC accepted.*

PEANUT SOFTWARE INC.
5045 North Academy Blvd. Suite 279
Colorado Springs, CO 80918
(719) 260-0598

CIRCLE NO. 366 ON READER SERVICE CARD

FIGHT PIRACY & PROTECT YOUR PROGRAM $$$'s!

Since 1986, companies worldwide have been choosing Az-Tech security products. If you demand the strongest protection available, why not choose one of these "proven leaders":

***EVERLOCK** Copy Protection
***EVERTRAK** Software Security
***EVERKEY** Hardware "Key" Software Security

Az-Tech Software, Inc.
305 East Franklin
Richmond, MO 64085
(800)227-0644 (816) 776-2700
Fax (816) 776-8398

CIRCLE NO. 368 ON READER SERVICE CARD

SORT FASTER THAN EVER BEFORE!

QSORT is the world's fastest sorting utility that will scream through sorting your ASCII or fixed-length binary files, using user-definable keys, lexicographically, AND MORE!

dQSORT will perform the same lightning-fast sorting operations on your dBASE data files. The awesome speed of dQSORT and QSORT will truly amaze you.

The QSORT/dQSORT package is available for only **$50.00.** Order the world's fastest sort utilities today!

System Enhancement Associates
21 New Street, Wayne, NJ 07470
(201) 473-5153
Visa and MC Accepted

sea

*dBASE is a trademark of Ashton-Tate

CIRCLE NO. 365 ON READER SERVICE CARD

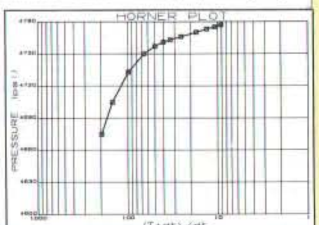
INGRAF 2.10 - Graphics Library

VIDEO, PRINTER & PLOTTER Graphics Library. Linear, LOG, Polar, Smith, Bar and Pie charts. Axes with tick mards & annotation. Multiple plots on a page. Marker and line types. Scalable fonts. Over 100 routines to make GRAPHICS programming easy on IBM-PC/XT/AT and compatibles.

Versions for: **C, FORTRAN Quick-BASIC and Pascal**

$295.00
Full Source code. No Royalties.

Sutrasoft
10506 Permian Dr.
Sugar Land, TX 77478
(713) 491-2088



CIRCLE NO. 367 ON READER SERVICE CARD

OPT-TECH SORT/MERGE

Extremely fast Sort/Merge/Select utility. Run as an MS-DOS command or CALL as a subroutine. Supports most languages and filetypes including Btrieve and dBase. Unlimited file sizes, multiple keys and much more! MS-DOS $**149.** OS/2, UNIX **$249.**

(702) 588-3737
Opt-Tech Data Processing
P.O. Box 678 - Zephyr Cove, NV 89448

CIRCLE NO. 369 ON READER SERVICE CARD

INSTALLATION TOOLKIT

Acomplete source toolkit of functions in C, Turbo Pascal or Microsoft Basic for DOS installation processes including windowing facilities, extensive hardware/system sensing, high performance data compression, readme file browsing. MediaBuilder tool automates building of compressed or uncompressed distribution media. Handles most of the drudgery of producing slick, professional installation processes, dealing with most errors and specific situations. Polite, intelligent modification of AUTOEXEC, CONFIG, WIN.INI and other files. Special facilities for installation of windows products. Designed for easy international translation. Includes INSTALIT and other special installers for various situations. No royalties.

$149 (DOS)

• Helpful Programmers, Inc. •
Central Bank Building, Suite 912
P.O. Box 16078, Huntsville, Alabama 35802-1627
Order Visa Master Card or mail, or call
800-448-4154

CIRCLE NO. 363 ON READER SERVICE CARD

STOP SOFTWARE PIRACY

With our Copy Protection Products. They really work and don't burden the honest user..

For Hardware Based Security (Parallel Port)
- STANDARD KEY TAG Unlimited Runs
- COUPON KEY TAG Limited Runs
- DURATION KEY TAG Limited (days)

For Disk Security
High Level Security
- PADLOCK II DISK • COUPON DISKS
- SAFEGUARD DISKS

For Hard Disk Protection
- HDCOPY

Low Level Security - User Installable Protection
- PC-PADLOCK

The market is filled with copy protection products which burden the user or simply don't work. we have over 3500 satisfied software firms utilizing our systems. The high-level fingerprint has not required an update in over 3 years.
Why should your valuable data or useful software program become available in the Public Domain?

Call or Write for more information.

GLENCO ENGINEERING INC.
SERVING THE SOFTWARE INDUSTRY SINCE 1979
721 W. Algonquin Rd. (312) 364-SOFT (7638)
Arlington Hts., IL 60005 FAX 364-7698 Telex 493-7109

CIRCLE NO. 370 ON READER SERVICE CARD

Programmer's MARKETPLACE

How to place your ad in the Programmer's Marketplace:

Carefully type your message or send camera-ready art. Each ad is 10 x 13 picas (15/8 x 23/16 inches). All ads must be prepaid. We accept checks, money orders, Visa, Mastercard, and American Express. Send your ad to: ***Programmer's Marketplace,*** *Dr. Dobb's Journal,* 501 Galveston Drive, Redwood City, CA 94063. For more information, contact: Glynn Mansfield at (415) 366-3600.

Source Code for C Developers

new gnu c + +	$150.00
new gnu c	$150.00
new gnu symbolic debugger	$45.00
ghostscript	$45.00
gas	$45.00
binutils	$30.00
bison	$30.00
grep	$15.00
PC Curses	$15.00
new PCCurses with Microsoft C 5.1 lib	$20.00

Specify MS-DOS or CPIO format
Call or write for catalog of latest offering

Networks
80-47 88 Road
Woodhaven, NY 11421 **(201) 206-0320**

CIRCLE NO. 371 ON READER SERVICE CARD

COMM-DRV
INTERRUPT DRIVEN SERIAL COMMUNICATIONS LIBRARY AND MS DOS DEVICE DRIVER

Make your serial communication application interface fast, robust, and error free through COMM-DRV. Free user and technical support.
Supports:
- Most multiport serial cards,
- Up to 25 ports (COM1-COM25), Up to 38400 baud,
- XON/XOFF and RTS/CTS/DSR/DTR handshaking,
- Polled and Interrupt driven I/O (Background Processing),
- Adjustable communication buffers,
- Sharing of several ports per IRQ,
- Running under multitasking systems like Desqview and Omniview. Transparent interface to DOS INT21', BIOS INT14H, C'S open(), read(), write(), close() and fast direct calls. Displays statistic on every port

Device Drivers and library **$39.95** **VISA/MC/COD/CHECK**
Device Drivers, library, and source code - **$89.95**
713-498-4832
Willies' Computer Software Co.
2470 S. Dairy Ashford Suite 188 Houston, Texas 77099

CIRCLE NO. 372 ON READER SERVICE CARD

The SPINDRIFT Library

At Last!! Now there's a way for the FORTRAN programmers to do the things they have always wanted to do: Execute other programs via CALL EXEC; direct control of the cursor with edit keys; WINDOWS on the screen for POP UP HELPs; (*) and (?) wildcard file searches; save/restore screen images; COLOR screens.

These are just a few of the features of the SPINDRIFT Library. Over 150 subroutines and functions in all, the SPINDRIFT Library includes DOS Interface (COPY, ERASE, MKDIR, FINDFILE, SYSTEM, etc), Variable Length Strings, Security Routines, Date/Time Routines, 10 SORT Routines, and much more!

Price $149 plus shipping/handling
Write for a DEMO DISK $5.00 credited toward purchase.
Specify your FORTRAN Compiler

Spindrift Laboratories, Ltd.
116 South Harvard Avenue
Arlington Heights, Illinois 60005
(312) 255-6909

CIRCLE NO. 373 ON READER SERVICE CARD

85,000 professional programmers will read this ad...

And you could have advertised to all of them for about 1/2-cent each.

Get your advertising message in the ***Programmer's Marketplace!***

Call Glynn Mansfield at (415) 366-3600

CIRCLE NO. 374 ON READER SERVICE CARD

Network API Library

For Microsoft and Borland C or Pascal. Compatible with IBM, 3Com, Banyan, and Novell networks. Includes DOS and NetBIOS network functions. Netware API's for login, mapping, print queues, IPX/SPX, etc. *New!* Portable IPC, RPC, and remote window routines. $99.95

Information Software Associates
P.O. Box 2582, Iowa City, IA 52244
Call for more info (319) 354-6804

CIRCLE NO. 375 ON READER SERVICE CARD

QuickBASIC File Indexing

Create B+ Tree file indexes in QuickBASIC with **Index Manager.**™ Access files randomly by full or partial key, or sorted forward or backward. One module performs all functions making **IM** very easy to use. Only indexes managed! You retain full control over your data. Assembler code and cache buffers make **IM** very fast. Demo on GEnie and CompuServ. Eight sample programs and 64 page Users Guide included. Only $59 (+ tax in CA). VISA/MC accepted.

CDP Consultants
1700 Circo Del Cielo Drive
El Cajon, CA 92020
(619) 440-6482

CIRCLE NO. 376 ON READER SERVICE CARD

FORTRAN PROGRAMMER?

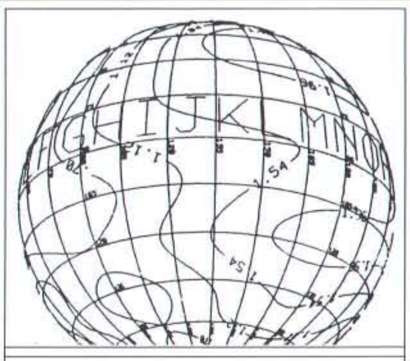
Now you can call 2-D and 3-D graphics routines within your FORTRAN program.

GRAFMATIC:	screen routines	$135.
PLOTMATIC:	plotter driver	$135.
PRINTMATIC:	printer driver	$135.

For the IBM PC, XT, AT & compatibles. *We support a variety of compilers, graphics bds., plotters and printers.*

MICROCOMPATIBLES
301 Prelude Drive, Silver Spring, MD 20901 USA
(301) 593-0683

CIRCLE NO. 377 ON READER SERVICE CARD



DYNAMICS GRAPHICS LIBRARIES

Save Development Time!
Contour, Bar, Pie, X-Y, Log, **Spherical** & 3D Fishnet graphs are autoscaling. Dynamic 3D motion of models. Fast polygon fill. Write one routine to support CGA/EGA/VGA/Herc. & Dot Matrix/Laser/Plotters at full resolution. *No Royalty!* For "C" QuickBASIC & FORTRAN. Starts at $140!

CHIRP TECHNICAL SERVICES
Call For Details! (619) 632-9510

BASIC to C

BAS_C v3.2 accepts BASICA, Quick BASIC v3, CBASIC86, or other different BASICs by adding new statements, functions (Customizer), generates structured, indented, scoped C, MS/Turbo/Lattice/Aztec C. All memory model, compile option are supported. C source code (Runtime Library) is included. Run on MSDOS, XENIX, UNIX. 95% conversion ratio. Demo disk. From $280.

GOTOLESS CONVERSIONS
P.O. Box 835910
Richardson, TX 75083
(214) 404-1404

CIRCLE NO. 378 ON READER SERVICE CARD

UNDERSTANDING C

6 hours of videotape gives the in-depth details of the ANSI C language. 200 screens of examples teach you to write like an experienced pro. Learn about structs, pointers, bit fiddling - all the details of C syntax. 100% satisfaction guaranteed! $59.95 (no extra charge for C.O.D.) Call now for info or C.O.D. order.

1-800-762-5515

Automation Education
2309 Royce Dr.
Arlington, Texas 76016-1232

CIRCLE NO. 379 ON READER SERVICE CARD

DEVELOPMENT LIBRARIES FOR PC PROGRAMMERS
Source always included!!!

AVAILABLE FOR PASCAL, ADA, MODULA-2 and C

product	per language	all languages
*BTree-core and disk versions	$42.95	$139.00
*Lists, stacks and queues	$34.95	$109.00
*Trees-binary and heaps	$42.95	$139.00
all packages	$95.00	$295.00

order direct: **REGAN SALES**
PO BOX 2204
BOTHELL, WA 98041-2204
(206) 820-2603

CIRCLE NO. 380 ON READER SERVICE CARD

Programmer's MARKETPLACE

Finally!

An Operating System which brings Real-Time response, multi-Tasking operation, Multi-User support, and Presentation Management to the DOS operating environment!

Introducing...

T/L Executive ™

The ideal software solution for programmers developing real-time, multi-tasking, DOS compatible application programs or enhancing existing DOS application programs.

Features:

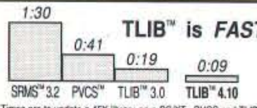
- Real-Time Kernel, Preemptive Task Scheduler;
- Fast, up to 12,820 task switches/sec (16Mz AT);
- Multi-Tasking, up to 255 prioritized tasks;
- Full DOS Compatibility;
- Runs on any PC-XT, AT, PS/2, or compatible;
- Efficient, requires only 40KB of RAM;
- Complete Intertask Communication capabilities;
- User Configurable Operating Parameters;
- Presentation Management Menus and Windows;
- Multi-User Support for up to 35 system users;
- Execute tasks in up to 8 MB of EMS RAM;
- NETBIOS compatible Networking capabilites;
- User Programmable Function Call Security;

For more information about T/L Executive, please call or write:

Talton/Louley Engineering

9550 Ridgehaven Court
San Diego, CA. 92123
(619) 565-6656

CIRCLE NO. 381 ON READER SERVICE CARD



TLIB™ is *FASTEST!*

Times are to update a 45K library on a PC/XT. PVCS and TLIB 3.0 are from Sept 87 *PC Tech Journal*. SRMS and TLIB 4.10 are later versions.

"TLIB is a great system" Jim Vallino, **PC Tech Journal**

Full-Featured Version Control for Software Professionals.

Check-in/out locking, branching, keywords, audit trail, wild-cards, file lists. Can merge parallel versions and undo intermediate revisions. **Mainframe compatible** deltas for Pansophic, ADR, IBM, Sperry. Integrated Landon Dyer PD **MAKE**. ***Just $99.95*** + $5 s/h. 5 station LAN license $299.95 + $5 s/h, call for other sizes. MS-DOS 2.x, 3.x, 4.x.

BURTON SYSTEMS SOFTWARE
PO Box 4156, Cary, NC 27519 **(919) 856-0475**

CIRCLE NO. 382 ON READER SERVICE CARD

HUGE Professional Shareware Libraries

Comprehensive, up-to-date, indexed collections of all PD/Shareware utilities for professional developers organized by specialty area.

Product	Disks/Files	Price	Product	Disks/Files	Price
dBase/dBase Compilers	50/1000	$99.50	Assembler	11/170	$49.50
AutoCAD	20/600	$99.50	Dos Utilities	20/200	$49.50
Turbo Pascal	19/200	$49.50	C	19/400	$49.50
Lotus 123	15/250	$49.50	Ventura DTP	21/170	$49.50
			Games	30/200	$49.50

Visa, MCard, Discover (inlcude exp. date!) COD, Corporate POs
EMS, 4505 Buckhurst Ct., Olney, MD 20832 USA
(301) 924-3594 Fax: (301) 963-2708
In France: (1)47763872

CIRCLE NO. 383 ON READER SERVICE CARD

PROGRAMMERS' SHAREWARE

Language sets. Each disk in these sets is achived with over 700K of utilities, toolkits, graphics, and windowing support. All include necessary documentation and most include source code.

- ☐ C(LIB) Microsoft and Turbo C, over 4.5 MB, 7 disks, $20
- ☐ (TP4) Turbo Pascal, 4.5 MB, 7 disks, $20
- ☐ (DBASE) dBASE III + Library, over 4.5 MB, 7 disks, $20
- ☐ (ASEM) Assembler Library, over 5 MB, 7 disks, $20
- ☐ (PROLO) Turbo Prolog Lib., over 3 MB, 5 disks, $15
- ☐ (BASIC) Turbo and QuickBASIC, over 3 MB, 5 disks, $15
- ☐ Windows Boss (LC07) Windows, Pull Down & Pop Up Menus, Data Entry etc. Supports Turbo C, Microsoft & others
- ☐ DeSmet C Compiler (LC15) Full C Compiler, LinKer, Assembled! Complete C! First Full C Compiler in Shareware!
- ☐ Integrated Basic Compiler (LB08) Full environment: Edit, Compile, Debug and Run all within the environment, On Line Help. Much like QuickBASIC™!
- ☐ Smart Disassembler (LA12) Smart Labeling Disassemble to disk, screen or printer.
- ☐ Programmer's Reference Disk (UT30) Huge amounts of valuable reference material
- ☐ TSR Tutorial (UT31) Complete tutorial with source code examples. More!

Free shipping! Toll-free ordering! 48-hour turnaround! Free 64-page catalogue!
VISA/MC!
Computer Solutions Northwest
P.O. Box 446
Benzonia, MI 49616
(800) 327-2540 (616) 325-2540

CIRCLE NO. 384 ON READER SERVICE CARD

'C' and 'C++' DOCUMENTATION TOOLS

Automated documentation for C and C++ programs:

- **C-CALL ($59)** creates graphic tree of caller/called functions and files-vs-functions table of contents.
- **C-CMT ($59)** creates/inserts comment-blocks for every function listing functions/identifiers used.
- **C-LIST ($39)** lists and action-diagrams, or reformats programs into standard formats.
- **C-REF ($49)** creates cross-reference of local/define/gobal/ parameter identifiers.
- **SPECIAL ($149)** All 4 programs plus FREE integrated C-DOC program. 30-day guarantee.

SOFTWARE BLACKSMITHS INC
6064 St. Ives Way
Mississauga, ONT, Canada. L5N-4M1
(416) 858-4466 **!! CALL and ORDER NOW !!**

CIRCLE NO. 385 ON READER SERVICE CARD

(continued from page 152)

BrainMaker v2.0, a system for designing, building, training, testing, and running neural networks, is now available from **California Scientific Software**. The company claims that version 2.0 is a major enhancement, and has such features as the NetMaker, which is a network generation and data manipulation program with spread-sheet style data display and the ability to perform arithmetic operations on data. In addition, BrainMaker now reads data from Lotus, dBase, Quattro, and Excel files; automatic numeric translation allows the display of large numbers; graphics post-processing of network results is supported; and it includes training and running algorithms that are as fast as 500,000 neural connections per second. Mark Lawrence, CSS president, told *DDJ* that "with Release 1, we knew we had a neat technology, and that our users would have to tell us what it was for. They told us it was mostly for financial forecasting, so now we've gone back and written it like it should have been."

This upgrade includes the *Introduction to Neural Networks*, an overview of the history and research of neural networks, and a user's guide and reference manual. The cost is $195, $95 for registered users. Requires a PC, PS/2, or compatible. Reader service no. 21.

California Scientific Software
160 E. Montecito Ave., #E
Sierra Madre, CA 91024
818-355-1094

Macintosh Allegro Common Lisp (CL), v. 1.3, an extended implementation of Common Lisp, is being shipped by **Apple Computer**. It supports all the features described in the Guy Steele text *Common Lisp: The Language*. The Macintosh Allegro CL can be used to develop stand-alone Macintosh applications and to port applications developed on other machines.

User interface components can be modified both interactively and under program control. Events are caught and dispatched by the Lisp run-time kernel. Windows are accessible as high-level objects, and can be created and closed with simple Lisp functions. Menus and dialog boxes are implemented as objects, as well. Fred, an integrated programmable editor, combines the capabilities of Emacs with the multiple-window, mouse-based editing style of the Mac. The Stand-Alone Application Generator produces ready-to-use Mac applications, which require a "nominal fee" license to distribute. System requirements include any Mac except the 512K, a second 800K disk drive, and

CODE MASTER

Bar Code Printing Can Now Be Part Of Your Programs With Code Master Libraries!

Accuracy • Versatiltiy • Quality

•Increase profits •Enhance productivity
•DOS, Unix, and OS/2 compatible
•Integrates with C, Clipper, more
•Supports most PC printers •Money back gurantee

Royalty Free -$389- Source Included
Call For Details!
1-800-552-2331

THE COMPUTER CONNECTION
P.O. Box 634 • Eau Claire, WI 54702
(715) 833-2331

CIRCLE NO. 405 ON READER SERVICE CARD

JLaser 5 . . . $399
Laser Printer Controller

Thinking of buying a
- LaserMaster
- Intel Visual Edge™
- Kofax board

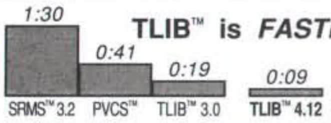
Wasting $$$

JLaser 5 gives you the functions of all three boards combined into one, *plus* EMS:
- Fast laser printing
- Halftones on a laser printer
- Group 4 file printing and display

TALL TREE SYSTEMS
2585 E. Bayshore Rd. • Palo Alto, CA 94303
(415) 493-1980 • FAX (415) 493-7639

Versions available for HP Series II and Canon LBP-4 laser printers
All products are trademarks of their respective companies
Artwork for this ad created with JLaser 5 and included software

CIRCLE NO. 411 ON READER SERVICE CARD



TLIB™ is *FASTEST!*

Times are to update a 45K library on a PC/XT. PVCS and TLIB 3.0 are from Sept 87 *PC Tech Journal*. SRMS and TLIB 4.12 are later versions.

TLIB™ is *BEST!*

"Do not be fooled by the fact that this is the least expensive of the five packages reviewed here - TLIB has features and power to spare" John Rex, **Computer Language**

"TLIB is a great system" J. Vallino, **PC Tech J**

• **Full-Featured Version Control for Software Professionals.** Check-in/out locking. Branching. Keywords. Wildcard and list-of-file support. Can merge parallel changes and undo intermediate revisions. Network and WORM support. **Mainframe compatible** deltas for Pansophic, ADR, IBM, Sperry. Integrated Landon Dyer PD **MAKE**.

MS-DOS 2.x, 3.x, 4.x ***Just $99.95*** + $5 s/h Visa/MC
5 station LAN license $299.95 + $5 s/h, call for other sizes

BURTON SYSTEMS SOFTWARE
PO Box 4156, Cary, NC 27519 **(919) 856-0475**

CIRCLE NO. 247 ON READER SERVICE CARD

Mac System Software, Version 6.0. It sells for $495. Reader service no. 28.
Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014
408-996-1010

MS-DOS Kermit, Version 3.0, is now available from the **Columbia University Center for Computing Activities**. The new features include transfer of text files in international character sets via a new Kermit protocol extension; emulation of the DEC VT320 terminal, including soft function keys and support for a wide variety of international character sets in any of the five standard PC code pages; and sliding window packet protocols for improved file transfer performance over public data networks and long distance satellite connections.

Version 3.0 also has expanded support for local area networks, and enhanced Tektronix graphics terminal emulation with VT340 extensions. Graphics screens may be saved in TIFF 5.0 for importation into such applications as WordPerfect, Pagemaker, and Ventura Publisher. This version was prepared by Joe R. Doupnik of Utah State University in cooperation with Columbia University. Reader service no. 29.
Kermit Distribution, Columbia Univ.
Center for Computing Activities
612 W. 115th St.
New York, NY 10025
212-854-3703

Stony Brook Professional Modula-2, Version 2.0, is now available from **Stony Brook Software**. This compiler package includes the QuickMod compiler and an optimizing compiler; development support for DOS, OS/2, and Microsoft Windows; the ability to interface with libraries written in C or other languages; the M16 debugger; an extensive run-time library; built-in multitasking; and the Stony Brook linker.

The Stony Brook environment gives you control over source file placement, keystroke macros, the ability to perform DOS commands, a cross-reference of module dependencies, interface to the symbolic debugger, and the ability to assemble foreign language modules. QuickMod complies with Wirth's definition of Modula-2, and supports symbol scoping and forward reference capabilities in one pass. Added are structured constants, array substrings, conditional compilation, type coercions, and set types containing up to 65,536 bits. The whole package retails for $295; the source code for the run-time library costs $150. Reader service no. 24.
Stony Brook Software
187 E. Wilbur R., Ste. 9
Thousand Oaks, CA 91360
800-624-7487 (US)
805-496-5837 (Calif. and International)

Books of Interest

NeuralSource, The Bibliographic Guide to Artificial Neural Networks, by Phillip Wasserman and Roberta Oetzel, is available from **Van Nostrand Reinhold**. This bibliography purports to be the most extensive collection of research information on neural nets. Periodicals, private reports, and books are included. Sells for $64.95. ISBN 0-442-23776-6.

Also by Wasserman is *Neural Computing, Theory and Practice.* This is an introduction to artificial neural networks for the nonspecialist. Assumes no math background beyond an undergraduate scientific education. Uses a step-by-step algorithmic approach to present commonly used network paradigms. $36.95, ISBN 0-442-20743-3. Reader service no. 30.
Van Nostrand Reinhold
P.O. Box 668
Florence, KY 41022-0668
606-525-6600

Elements of Functional Programming by Chris Reade has been published by **Addison-Wesley**. Covers the concepts and techniques used in modern functional programming languages, as well as support for abstraction, programming with lists, new types, abstract data types and modules, lazy and eager evaluation, and implementation techniques. Hardback edition costs $37.75, ISBN 0-201-12915-9. Reader service no. 31.
Addison-Wesley
Reading, MA 01867
617-944-3700

The *COSMIC Software Catalog 1990 Edition* is available from the **University of Georgia**. It is a comprehensive listing of program abstracts describing all available NASA computer programs. You can purchase it in book form ($25), on microcomputer diskette ($30), on magnetic tape ($50), or on microfiche ($10). The catalog cross-indexes over 1200 computer programs, in areas such as aerodynamics, reliability, composites, heat transfer, artificial intelligence, and structural analysis. Reader service. no. 32.
COSMIC
University of Georgia
382 E. Broad St.
Athens, GA 30602
404-542-4807

(continued on page 125)

EXPANDED SELECTION...

When we started Programmer's Connection in 1984, we dedicated ourselves to providing the highest level of quality service possible. As a result, we've become the world's leading independent dealer of computer products for IBM personal computers. We invite you to call us today and experience our quality service for yourself.

EXPANDED PRODUCT LINE. We have always provided you with the largest selection of software development tools. Now, to service all of your personal computer needs, we have expanded our product line to include popular business applications and hardware products. So, call today and find out why Programmer's Connection is your one-stop connection for personal computer products.

INFORMATIVE BUYER'S GUIDE. *The CONNECTION*, our FREE comprehensive Buyer's Guide and Catalog, contains prices and up-to-date descriptions of over 1200 products by more than 440 manufacturers. Each description includes major product features, special requirements, version numbers, diskette sizes, return policies, and other information. Plus there are interesting articles by leaders in the programming industry, New Product Spotlights, Technical Questions & Answers, and much more.

To Get Your FREE Copy: 1)use the reader service card provided by this journal; 2)mail us a card or letter with your name and address; or 3)use one of our convenient toll-free telephone numbers.

386 products

	List	Ours
C Network Compiler/386 *by Novell*	995	798
F77L-EM/32 *by Lahey Computer*	895	779
NDP C-386 *by MicroWay* ... *New*	895	799
QEMM-386 *Expanded Memory Mgr, by Quarterdeck*	100	65
WATCOM C7.0/386 *by WATCOM Group*	895	639

assembly language

ASMFlow *by Quantasm Corporation*	100	78
BIOS Pre-Processor *by V Communications*	140	109
Microsoft Macro Assembler *by Microsoft*	150	105
Re:Source *by Genesoft Corporation* ... *New*	150	119

blaise products

ASYNCH PLUS	189	134
C ASYNCH MANAGER	189	134
C TOOLS PLUS/6.0	149	99
POWER SCREEN *Supports C, Pascal and BASIC*	149	98
POWER SEARCH	149	99
Turbo C TOOLS/2.0	149	99
VIEW232 *Serial Data Analyzer*	189	149

c language

Lattice C Compiler *DOS & OS/2, by Lattice*	250	169
Microsoft C Professional Development Sys	495	339
TopSpeed C DOS Standard *by Jensen & Partners*	199	158
Turbo C 2.0 *by Borland*	150	99
Turbo C 2.0 Professional *by Borland*	250	169
WATCOM C7.0 *by WATCOM Group*	395	309

c utilities

BAS_C *by Gotoless Conversion*	280	249
C-DATA MANAGER *by Database Technologies*	300	239
C_talk *by CNS*	150	135
CODAN *by Implements* ... *New*	395	349
CODE MASTER *by The Computer Connection* ... *New*	289	239
Graphics-MENU *for Borland BGI, by Island Sys*	99	79
for MetaWINDOW	149	119
Heap Expander *by The Tool Makers*	80	72
PC Lint *by Gimpel Software* ... *New Version*	139	99
PRO-C *by Vestronix*	399	299
Sherlock *Debugger for C by Edward K. Ream*	195	174
SLATE *by Symmetry Group*	299	239
XVT *for DOS & OS/2 Character Displays, by GSS*	395	329
for Windows, PM or Mac Character Displays	595	498

c++ language & utilities

C++ 2.0 Compiler *by ZORTECH*	200	179
C++ 2.0 Debugger *by ZORTECH* ... *New*	150	129
C++ 2.0 Developers Edition *by ZORTECH*	450	399
C++ 2.0 Tools *by ZORTECH*	150	135
C++ 2.0 Video *by ZORTECH* ... *New*	500	429
Guidelines C++ *by Guidelines Software*	395	339
INTEK C++ *by INTEK* ... *New*	495	449

database management

Clipper *by Nantucket*	795	499
FORCE *dBASE Compiler, by Sophco*	695	559
WinTrieve *by Whitewater Group*	395	338

file management utilities

AccSys *for dBASE or Paradox,by Copia Int'l*	395	348
Code Base 4 *by SEQUITER Software*	295	239
FairCom Toolbox Professional *by FairCom*	1095	775
c-tree File Handler	395	278
r-tree Report Generator	295	198

graphic utilities

GRAFMATIC *by Microcompatibles*	135	119
GraphiC *by Scientific Endeavors*	395	318
Graphics Development Toolkit *by GSS*	595	498
Menuet *by Ithaca Street Software*	325	259
MetaWINDOW *No Royalties, by Metagraphics*	250	153
PCX Effects *with Source, by GENUS*	199	159
PCX Programmer's Toolkit *w/Source, by GENUS*	495	399
PCX Text *with Source, by GENUS*	299	249

greenleaf products

Greenleaf Business MathLib	395	278
Greenleaf Comm Library	299	165
Greenleaf DataWindows *Includes Source Code*	395	223
for OS/2	395	223
Greenleaf Functions	229	143
Greenleaf MakeForm-DOS	125	93
Greenleaf SuperFunctions	299	203
Greenleaf ViewComm	549	359

program mgmt utilities

PolyMake *UNIX-like Make, by Sage/Polytron*	149	124
PVCS Professional *by Sage/Polytron*	495	334
TLIB *Version Control System, by Burton*	100	83

screen management utilities

C-scape *w/Look & Feel, by Oakland* ... *New Version*	499	475
C Windows Toolkit *by Magna Carta Software*	100	79
Curses *by Aspen Scientific*	159	129
JAM *Application Manager, by JYACC*	595	463
Vitamin C *by Creative Programming*	225	163
VC Screen *Forms Designer*	149	113
Vitamin C for OS/2 *by Creative Programming*	345	278

south mountain products

C Utility Library	249	148
Essential B-Tree *with Source Code*	199	148
Essential Communications	329	199
Essential Graphics *with Source Code*	1398	999
Hold Everything	129	99
/*resident_C*/ *with Source Code* ... *New Version*	249	189

text editors

Brief *from Solution Systems* ... *Version 3.0*	199	SAVE
Brief for OS/2 *by UnderWare* ... *Version 3.0*	199	139
Epsilon *Emacs-like editor by Lugaru*	195	138
KEDIT *by Mansfield Software Group*	150	119
Vedit Plus *by CompuView*	185	159

turbo pascal utilities

B-Tree Filer *by TurboPower Software*	125	99
GraphLink Printer Graphics Toolkit *by Visitech*	125	109
Inside! Turbo Pascal *by Paradigm Systems*	125	109
Object Professional 1.0 *by TurboPower Software*	150	119
Turbo Pascal 5.5 *by Borland*	150	99
Turbo Pascal 5.5 Professional *by Borland*	250	169
Turbo Professional 5.0 *by TurboPower Software*	125	99
TurboGeometry Library *by Disk Software*	200	159

other languages

Actor *by Whitewater Group*	695	565
F77L-EM/16 *by Lahey Computer*	695	599
Microsoft BASIC Professional Develop Sys	495	339
PowerBASIC *by SPECTRA Publishing* ... *New*	130	119
Professional Modula-2 *by StonyBrook Software*	295	239
Prograph *for Mac,by TGS Systems* ... *New*	195	179
Smalltalk/V PM *by Digitalk*	500	399
TopSpeed MODULA-2 Standard *by JPI*	199	158

other products

AT BiosKit *by Annabooks* ... *New*	199	179
Batcom *by Wenham Software*	60	55
Berkeley Utilities *by OPENetwork*	125	109
DELTA *by OPENetwork*	79	69
Desqview *by Quarterdeck*	130	79
Dovetail *by Dovetail Communications* ... *New*	1495	1089
INSTALL *by Knowledge Dynamics* ... *New Version*	250	179
Intel Math CoProcessors *Call for Pricing*		
Opt-Tech Sort *by Opt-Tech Data Proc*	149	118
Overlay Toolkit *Small Model, by Lloyd Bush*	395	369
Periscope *Debugger, all Models by Periscope*	CALL	CALL
Plink86+ *overlay linker, by Sage/Polytron*	495	398
PromKit *by Annabooks* ... *New*	179	159
QNX Operating System *by Quantum Software* ... *New*	450	389

PROGRAMMER'S QUALITY INTEGRITY Service SUPPORT PRICE CONNECTION ®

PROGRAMMER'S CONNECTION

Order Processing Department • 7249 Whipple Avenue NW • North Canton, Ohio 44720 • USA

FREE SHIPPING. Orders within the USA *(lower 48 states only)* are shipped FREE via UPS Ground. Call for APO, FPO, PAL, and express shipping rates.

NO CREDIT CARD CHARGE. VISA, MasterCard and Discover Card are accepted at no extra cost. Your card is charged when your order is shipped. Mail orders please include expiration date and authorized signature.

NO COD OR PO FEE. CODs and Purchase Orders are accepted at no extra cost. No personal checks are accepted on COD orders. POs with net 30-day terms (with initial minimum order of $100) are available to qualified US accounts only.

NO SALES TAX. Orders outside of Ohio are not charged sales tax. Ohio customers please add 5% Ohio tax or provide proof of tax-exemption.

30-DAY RETURN POLICY. Most of our products come with a 30-day documentation evaluation period or a 30-day return privilege. Note that some manufacturers restrict us from offering return privileges on their products. Please consult our Buyer's Guide or call for more information.

SOUND ADVICE. Our knowledgeable technical staff can answer technical questions, assist in comparing products and send you detailed product information tailored to your needs.

INTERNATIONAL ORDERS. Shipping & Handling charges for International and Canadian orders are based on product weight. Rates are published in our Buyer's Guide (if you don't have a copy, call or write for the exact cost). All payments must be made with US funds drawn on a US bank. Please include your telephone number and FAX number (if any). Due to manufacturer and/or government restrictions, we cannot ship to some countries.

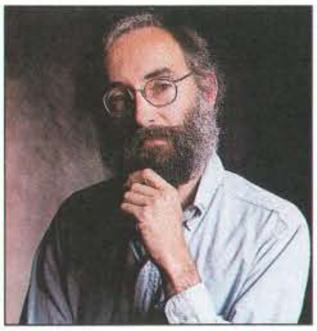
MAIL ORDERS. Please include your telephone number and complete street address on all orders. Specify computer, operating system, diskette size, and applicable compiler or hardware interface(s).

USA 800-336-1166

Canada	800-225-1166
International	216-494-3781
FAX	216-494-5260

Business Hours: 8:30 AM to 8:00 PM Eastern Time Monday – Friday
Prices, availability, terms and conditions are subject to change

Junk Customers



2/11/90: Investment banking firm Drexel Burnham Lambert, home of the junk bond phenomenon, informs its employees that it is filing for bankruptcy.
2/12/90: My cousin Corbett launches his program for software developers who can't afford the skyrocketing costs of software marketing: Junk Customers.

Corbett was concerned about the software developer who can't afford to run large ads in the major computer magazines and can't afford to rent lists of prospects. Some magazines and some lists will result in more responses and purchases than others, of course, and it was while trying to come up with a new measure of the value of these sources that Corbett hit on the secret, which he calls "Junk Customers."

He took his inspiration from Mike Milken, the Drexel Burnham Lambert employee who made such a splash with junk bonds. Milken noticed that there were a lot of companies that had to go to the bank when they wanted money. This was bad, he realized. The companies tried issuing bonds to get investors to put up money, but investment firms such as Drexel et al. steered investors away from these bonds, labeling them "low value." This meant that there was a higher probability that the companies would fail to pay up — go out of business or whatever — than was the case with so-called high-quality bonds. The companies tried to make their bonds appealing by increasing the yield —what you get back for your investment — and Mike Milken saw this as a good deal. He began helping his clients to buy a broad selection of these high-yield, low-value bonds, starting what became, at its peak, a $200 billion market.

Corbett has come up with a similar plan for software marketing. (The plan is completely general, but his loyalty is to the software development community. He wants you to have it.) He defines the yield of a source of prospects — a list of names or a page of advertising — as the inverse of the CPM (ad sales jargon for "cost per thousand"). Yield is how many names you get for a buck. He defines the value of a source as how well the source will pull — how likely each name is to result in a sale. The trick, as with junk bonds, is to develop a varied portfolio of high-yield, low-value sources.

Identifying a truly low-value source is tricky. It can't just be a source that is ill-suited to your needs; such a source might be able to get a lot of money from someone else. To ensure that the yield can be made high enough, this must be a source ill-suited to anyone's needs, a publication or list poorly suited to any commercial advertising or name rental purpose. Then there is another problem in dealing with low-quality sources: You'll need a lot of them. The low quality translates into few responses from any one source, and the overhead of dealing with hundreds of such companies can easily eat up any gains.

Corbett thinks he has found the single correct answer to the Junk Customers challenge, and is generously allowing me to pass it on to you: Church newsletters. Every community has a church, every church has a newsletter, and every church belongs to some large national or international organization capable of serving as a central clearing house for ad sales or list rental. The nonprofit status of churches and their general, noncommercial slant makes a church newsletter an exceptionally low-value source, Corbett maintains.

He sees an intriguing wrinkle to the idea of church newsletter subscribers as prospects for software sales. Current wisdom says that you should look for software prospects among owners of computers. The church newsletter subscribers will include many who do not own a computer, apparently nonprospects almost by definition. But any good marketer knows to mistrust such self-fulfilling predictions, and to ask the positive question, why would this person want my product? In this case, the answer is surprisingly obvious. The industry has been doing it backwards!

Consider: It is much easier to ease a potential customer into a new product category with a small purchase than with a large one. One of the reasons many people cite for not buying a PC is that they don't know how to justify spending over a thousand dollars. So they buy a Nintendo instead. These people could be buying your CAD package.

Consider: Anyone who has ever thought about buying a PC has heard the advice, "Decide what software you want to run, and then buy the PC that runs that software." You've probably given that advice, but did you listen to what you were really saying?

Consider this pitch: For less than the cost of a Nintendo, you can own the most powerful CAD package in the known universe. Now you, too, can design microprocessor circuits, draft plans for a new house on the coast, develop a new art form, make your own clothes. Required Silicon Graphics IRIS workstation must be purchased separately.

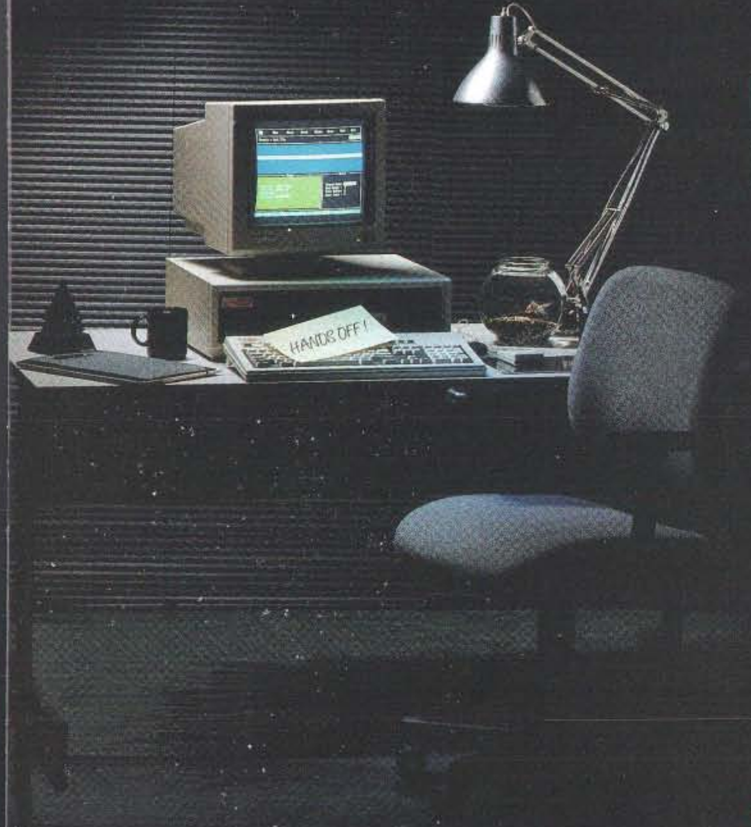
Remember, you read it here first.

Michael Swaine

Michael Swaine
editor-at-large

A T R O N

NOW YOUR SOFTWARE CAN TEST ITSELF.



NOW AVAILABLE IN VGA

AND PS2!!

Your customers expect software that works. All the time. The key to software quality is exhaustive testing. It's also an engineer's worst nightmare. But it doesn't have to be. Because now you can automate your software testing.

Introducing the Atron Evaluator. The first and only non-intrusive automated PC-based software testing tool.

The Atron Evaluator automatically runs your software regression testing programs. All of them. All day. All night. Giving you thoroughly tested, higher quality software.

The Atron Evaluator is hardware-based. And since it's non-intrusive, software behavior is tested without the risk of alteration. Once your tests have run, you can refer to automatically generated test reports to double-check test results.

The Atron Evaluator saves time. And time makes you money. Development cycles are shortened, so your software gets to market sooner. And while your test programs are running, you can be more productive. Start a new project. Or go home.

For more information about the Atron Evaluator, call us at **1-800-283-5933.** And put an end to your worst nightmares. Automatically.

atron
A Division of CADRE Technologies

Saratoga Office Center
12950 Saratoga Avenue
Saratoga, California 95070

In Europe, contact:

Elverex Limited, Enterprise House
Plassey Technology Park, Limerick, Ireland
Phone: 061-338177

QA Training Limited, Cecily Hill Castle
Cirencester, Gloucestershire, GL7 2EF, England
Phone: (0285) 5888

BORLAND'S TURBO C PROFESSIONAL 2.

CORPORATE SATISFACTION POLL · TOP RATED · PC WEEK · 1989

PC WEEK POLL: C COMPILERS

	Overall Weighted Score	Overall Reliability	Complete of Command Descript.	Overall Perform.	Complete & Organiz. Document.	Document Clarity	Compiling Process Efficiency	Product Support Quality	Value Relative To Cost	Product Support Access.
Turbo C 2.0 (Borland International)	81	87	79	84	77	78	86	72	70	93
C Optimizing Compiler 5.1 (Microsoft Corp.)	76	83	80	81	78	74	76	68	67	70
C++ 1.07 (Zortech Inc.)	66	68	64	71	63	63	69	60	58	76

"Microsoft was No. 1, but they have been unseated by Borland." PC Week, *May 8, 1989*

PC WEEK POLL: SOFTWARE DEBUGGERS

	Overall Weighted Score	Overall Reliability	Effective. Programmer Interface	Document. Clarity	Complete. Command Descript.	Complete. & Organize. Document	Overall Perform.	Integration Within Programming Environment	C Compiler Compatibility	Product Support Quality	Product Support Access	Value Relative To Cost
Turbo Debugger 1.0 (Borland International)	84	89	90	81	81	81	89	88	81	73	72	93
Codeview 2.2 (Microsoft Corp.)	73	80	71	72	74	74	74	74	78	67	64	72

"Borland's Debugger outshines Microsoft's Codeview." PC Week, *May 15, 1989*

It's two winners in one.

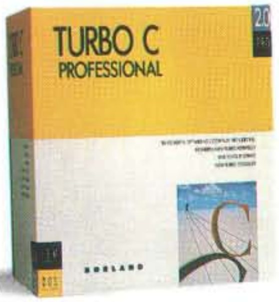
Turbo C,® the core of Turbo C Professional, was the outright winner in *PC Week*'s Poll of Corporate Satisfaction on C compilers. Overall, Borland won with 81. Microsoft® placed second.

Turbo Debugger,® also included in Turbo C Professional, was the outright winner in EVERY category in *PC Week*'s Poll Of Corporate Satisfaction on Debuggers. And, once again, we topped the score with 84, overall. Microsoft came in second-best, 11 points behind.

Get Borland's Turbo C Professional and get the best of both worlds: our top-rated C compiler and our top-rated Debugger.

Call **(800) 345-2888*** and we'll send you both *PC Week* polls and technical specifications on Turbo C and Turbo Debugger.

Turbo C Professional includes both Turbo C 2.0 and Turbo Assembler® & Debugger.



B O R L A N D

*From Canada, Call (408) 848-4391. Reprinted from PC Week, May 8, 1989; PC Week, May 15, 1989. Copyright ©1989 Ziff Communications Company. Turbo C, Turbo Debugger, and Turbo Assembler are registered trademarks of Borland International. Copyright ©1989 Borland International, Inc. All rights reserved. BI 1328

CODE: MC58

CIRCLE NO. 95 ON READER SERVICE CARD